


Операционные системы реального времени для интеллектуальных информационных систем

Лекция 6

Вопросы

1. Обзор основных направлений развития операционных систем реального времени.
2. Операционная система Spox.
3. Операционная система Multiprox.
4. Операционная система VCOS.
5. Операционная система DEASY.
6. Операционная система UNIX.
7. Операционная система OSF/1 и DCE.
8. Операционная система VAX/VMS.

Обзор основных направлений развития ОС РВ

Операционные системы реального времени (ОСРВ) используются в тех случаях, когда работоспособность обслуживаемой ими цифровой системы определяется не только результатом обработки поступившей информации, но и длительностью времени получения результата.

Современные ОСРВ должны удовлетворять ряду противоречивых требований: малый объем, достаточный для размещения в резидентной памяти системы; малое время отклика; реализация многозадачного режима с гибким механизмом приоритетов, наличие сервисных функций и средств поддержки для разработки прикладных программ и ряд других. В настоящее время разработчику систем предлагается ряд ОСРВ, имеющих различные характеристики и прошедших апробацию в многочисленных областях применения, что позволяет ему найти компромиссное решение для выполнения поставленной задачи. Наиболее часто в системах на базе микропроцессоров и микроконтроллеров фирмы Motorola используются следующие ОСРВ: OS-9 фирмы Microware Systems; VxWorks фирмы WindRiver Systems; LynxOS фирмы Lynx Real-Time Systems; pSOS+ фирмы Integrated Systems; QNX фирмы Quantum Software Systems; VRTX/OS 3.0 фирмы Ready Systems; Nucleus фирмы Accelerated Technology; RTXC фирмы Embedded System Products; OSE фирмы Enea Data, Precise/MQX фирмы Intermetrics Microsystems Software; VMEexec фирмы Motorola.

Подразделяют ОСРВ на два класса - системы "жесткого" и "мягкого" реального времени (РВ). Системы "жесткого" РВ имеют минимальные объем и время отклика, но обладают ограниченными сервисными средствами. Типичным примером ОСРВ этого класса служит VMEexec. Системы "мягкого" РВ требуют большего объема памяти, имеют более длительное время отклика, но удовлетворяют широкому спектру требований пользователя по режиму обслуживания задач, уровню предоставляемого сервиса. Примером такой ОСРВ может служить OS-9/9000..

Обзор основных направлений развития ОС РВ

Однако для современных ОСРВ данная классификация является весьма условной. Ряд ОСРВ, относящихся к классу "жестких", имеют средства интерфейса, которые позволяют, в случае необходимости, использовать высокоэффективные отладчики или интегрированные среды разработки, обеспечивая, таким образом, пользователя набором средств поддержки программирования-отладки систем.

Например, VMEexec может использоваться совместно с интегрированной средой MULTI фирмы GreenHills Software, VxWorks с интегрированной средой Tornado, в составе которой поставляются отладчик CrossWind и GNU-компиляторы фирмы Cygnus Support, VRTX и pSOS с отладчиком XRAY и компиляторами фирмы Microtec Research. С другой стороны, системы "мягкого" РВ реализуются по модульному принципу, что позволяет использовать только те средства, которые необходимы в данном приложении. В результате для конкретного применения достигается существенное сокращение объема необходимой памяти и времени отклика. Например, для ядра OS9/9000 время отклика не превышает 20 мкс (для VMEexec, VxWorks, pSOS - менее 10 мкс), что является вполне достаточным для многих приложений.

Для создания многопроцессорных систем, работающих в режиме реального времени (РВ), необходимо базовое программное обеспечение, а именно операционная система. ПО этого направления делится на две большие группы. К первой группе можно отнести небольшие модули, загружаемые на ЦОС-процессоре, а также библиотеки подпрограмм для основного процессора, позволяющие реализовать обмен данными. ЦОС-процессор в такой системе является подчиненным процессором, управляемым основным (host-процессором). Организация функций систем РВ основана на обработке прерываний и механизме обмена сообщениями. Главное достоинство этих систем – небольшая цена. К системам этого типа можно отнести VCOS и DEASY.

Вторая группа операционных систем – это операционные системы реального времени типа Spox или Multiprox. Цена этих систем составляет порядка 20-40 тыс. долларов, но возможности их значительно выше. Операционная система Spox фирмы Spectron Microsystems – одна из ведущих систем, используемых в системах РВ для различных применений, в том числе на платформах с модулями ЦОС. Spox – это специализированная операционная система реального времени, создающая операционное окружение для приложений по обработке данных в реальном режиме.

Другая система – Multiprox фирмы Comdisco. Multiprox – это система разработки, которая позволяет инженерам графически формировать приложения и разделять ЦОС-задачи для нескольких ЦОС-процессоров.

Операционная система SPOX

SPOX (создана фирмой Spectron в 1987 г.) является операционной системой, которая структурирована для обработки сигналов и приложений с интенсивной математикой. Это высокоуровневое окружение для приложений имеет простые в использовании свойства, включая независимый от устройств ввод-вывод, удобные установки процессора и интерфейс с основной машиной (имеется в виду основная ОС). Обеспечивает объектно-ориентированную модель для ЦОС и математической обработки.

Однородные встраиваемые системы. Процессор ЦОС играет роль как общецелевого, так и специализированного процессора. Со SPOX ЦОС-процессор одновременно выполняет алгоритмы обработки сигналов вместе со сложным контролем по связи задач, прежде выполняемых на специализированных контроллерах. SPOX поддерживает многопроцессорность.

Разнородные встраиваемые системы. Встроенные компьютерные системы РВ с полными чертами ОС (например, VXWorks, OS-9, LynxOS) выполнены на основе ЦОС-подсистем. Это традиционная конфигурация, где ЦОС-подсистема прибавляется к встраиваемой компьютерной системе. Spectron предлагает сбалансированный подход, объединяющий традиционные встраиваемые компьютерные системы (используемые в промышленности) и DSP. Это открывает новый диапазон возможностей для проектирования встроенного управления.

Компьютерные интегрированные системы. В данной конфигурации рабочие станции контролируют ЦОС-подсистемы. Приложение ЦОС запускается в привычном интерактивном окружении (MS-Winows, Unix, DOS), выполняя приложение как тест или измерение, мониторинг контроля процесса, медицинские представления, сбор данных. Здесь приложение имеет ресурсы как основной, так и ЦОС-системы, действуя под управлением рабочей станции.

Мультимедиа системы. Компьютер требует мощных вычислительных затрат по воспроизведению мультимедийных приложений, что соответствует задачам ЦОС: аудиозапись и воспроизведение, видео в реальном режиме, распознавание речи, синтез звука, телекоммуникационные функции, такие, как факс, модем. ЦОС-модуль размещается либо на материнской плате, либо на дополнительной плате, а SPOX усиливает возможности мультимедиа в привычном пользовательском окружении.

SPOX поддерживает высокопроизводительную многозадачность, обработку прерываний, управление памятью, ввод-вывод в реальном времени и большой набор функций по обеспечению взаимосвязи между задачами и процессорами; имеет математическую и специализированную ЦОС-библиотеку функций; поддерживает модель ООП над векторами, матрицами и фильтрами. Для обеспечения этого имеется символьный отладчик и компиляторы

Операционные системы MULTIPROX, VCOS

Второе направление в развитии ПО – это Multiprox фирмы Comdisco, – пакет, который является новым выбором SPW (Signal Processing Workstation).

Используя инструментальное множество, инженеры могут определять ЦОС-приложения графически, используя графические объекты, которые представляют компоненты ЦОС- обработки. Multiprox позволяет инженерам выделять разделы среди потоков данных, рисовать диаграмму течения данных и определять порции, работающие на разных процессорах.

Дополнительно SPW-инструментальное множество и Multiprox автоматически преобразуют диаграммы к процессорно-зависимому Си-коду и встраивают в ПО связи или межпроцессорную коммуникацию, чтобы передавать данные от одного процессора другому. Диаграммы потоков данных преобразуются в Си-программу, содержащую подпрограммы, некоторые из которых написаны на ассемблере и вручную оптимизированы. Таким образом, инженер может использовать инструменты, чтобы распределять высокоуровневое ПО на различные процессоры или смешивать процессоры.

VCOS (Visible Caching Oparating System) делает процессоры ЦОС сопроцессорами. VCOS – переносимая, многозадачная и многопроцессорная операционная система реального времени. VCAS (VCOS Application Server) – резидентная на host-системе программа, загружает и связывает ЦОС-задачи и обеспечивает управление памятью и буферизацию ввода-вывода между host-и ЦОС-процессорами.

VCOS является “минимальной” операционной системой – она занимает менее 400 32-разрядных слов в памяти процессора. ОС использует память host-системы для запоминания программы и данных. Она использует ее как ресурс, чтобы кэшировать данные и код для более быстрой обработки на процессоре. VCOS является “подчиненной” по отношению к host ОС, которая располагает и контролирует VCOS структуры данных, избегая таким образом соперничества между подсистемами при доступе к памяти. VCOS выполняется в высоко приоритетном режиме.

VCOS поставляется вместе с полной библиотекой ЦОС-функций для мультимедийных приложений. Эти приложения включают V.32 модем, V.29 FAX модем, видеозапись, обработку речи, графику, функции сжатия

Операционная система DEASY

Операционная система DEASY предназначена для разработки, отладки и выполнения программ ЦОС для процессорных модулей DSP3x фирмы "Инструментальные системы". DEASY – однозадачная система реального времени. Ни одна из системных функций не блокирует обработку внешних событий (прерываний процессора) на величину более 500 нс.

Основная концепция при разработке программ с использованием DEASY заключается в том, что вокруг сигнального процессора создается виртуальная операционная среда, позволяющая ему играть роль центрального процессора для всего вычислительного комплекса, построенного на базе ПК. Таким образом, обеспечивается прозрачный доступ процессора ЦОС ко всем ресурсам ПК, включая экран монитора, клавиатуру, дисковые устройства, память, порты ввода-вывода и т.д. При этом облегчается процесс переноса программ из другой среды программирования и быстрое прототипирование программ обработки сигналов с использованием традиционных способов.

ОС DEASY включает набор библиотек и утилит для составления и отладки прикладных программ ЦОС. Библиотека System.a30 содержит набор функций для управления процессором TMS320C30 и доступа к его внутренним регистрам из Си-программ. Библиотека Host.lib состоит из функций для инициализаций, загрузки, управления и обмена информацией между платой ЦОС и ПК. Библиотека Deasy.a30 содержит набор функций-аналогов библиотеки компилятора Borland C и используется для "прозрачного" доступа к ресурсам ПК из прикладной программы, выполняемой на плате ЦОС. Библиотека Bgi.a30 включает в себя функции-аналоги графической библиотеки компилятора Boland C и используется для доступа к графическим ресурсам IBM PC.

Исполняющая среда Deasy.exe предназначена для загрузки и выполнения прикладных программ, составленных с использованием приведенных выше библиотек. Она также содержит простейший диалоговый монитор интерактивного взаимодействия с платой ЦОС. Символьный отладчик Kg30.exe, или символьный отладчик Cg30.exe, предназначен для отладки прикладных программ, выполняемых на плате ЦОС.

Для организации работы многозадачного режима необходимо обмениваться сообщениями, которые отображают текущее состояние работы программ. Например, после окончания вычислений на одном процессоре (DSP) необходимо сообщить центральному процессору (CPU) о том, что он может забрать данные. При пересылке данных с CPU на DSP необходимо сообщить DSP, что он должен производить расчет. Библиотеки системных функций System и Host позволяют организовать обмен сообщениями и данными. Библиотека System предназначена для компоновки с программами пользователя, написанными на языке Си или на Ассемблере для процессора TMS320C30. Библиотека Host предназначена для управления платой DSP со стороны ПК из программы

Операционная система UNIX

Операционная система UNIX представляет собой многозадачную, многопользовательскую опера-ционную систему и является в настоящее время одной из наиболее распространенных в мире.

UNIX непрерывно развивалась и существует в нескольких модификациях. Основными распространителями являются компании AT&T Bell Laboratories и Berkeley Software Distribution. В UNIX были введены средства, на базе которых разработаны другие ОС и важные коммуникационные интерфейсы, в частности протокол TCP/IP и протокол пользовательского терминала X Window.

UNIX состоит из небольшого ядра, управляющего системными ресурсами (процессор, память и ввод/вывод), а остальная часть процедур ОС работают как пользовательские процессы. Типичная операционная система UNIX содержит 10000-20000 строк на языке С и 1000-2000 строк машинно-ориентированных программ на ассемблере, которые разрабатываются отдельно для каждой аппаратной платформы. Ядро представляет собой единую резидентную программу размером от 100 Кб до 1 Мб в зависимости от платформы и выполняемых функций. При переносе системы UNIX на конкретную платформу надо переписать только машинно-зависимую часть ядра, поэтому UNIX может работать на многих аппаратных платформах с идентичным системным интерфейсом.

Первоначально система UNIX была разработана как многопользовательская, а не для приложений РВ. Из-за того, что подпрограммы ОС работают как пользовательские процессы, но с наивысшим приоритетом, назначенным системой, невозможно прерывать также те системные вызовы, выполнение которых занимает много времени, что увеличивает время реакции системы и является существенным недостатком. В UNIX используется довольно сложное описание контекста, что увеличивает время переключения процессов. Стандартно процессы в UNIX протекают с разделением времени. Для того чтобы дать всем процессам возможность исполняться, применяется динамическое распределение приоритетов. Важной особенностью, реализованной в UNIX, является одинаковая трактовка всех устройств. Внешние устройства ввода/вывода рассматриваются как файлы. Это существенно упрощает программы, требующие определенной гибкости. Это также важно и с точки зрения машинной независимости программ.

Общим недостатком UNIX является его недружественный пользовательский интерфейс. Положительной особенностью команд UNIX является то, что благодаря стандартизации ввода/вывода и механизму каналов несколько команд можно объединить в одной строке, причем выход одной команды является входом следующей. Хотя в начале UNIX была многозадачной операционной системой, не предназначенной для работы в реальном времени, стала очевидной необходимость ее адаптации и к задачам реального времени. Поэтому новые версии поддерживают такие функциональные элементы систем реального времени, как семафоры, разделяемую память, обмен сигналами между процессами,

Операционная система OSF/1 и DCE

Первоначальные версии UNIX не требовали лицензий и были доступны практически всем для свободного использования, что отчасти объясняет популярность этой системы.1

При выпуске System V компания AT&T решила распространять ее только с оплатой лицензий. Некоторые наиболее крупные производители ЭВМ - Digital, Equipment, Hewlett Packard, IBM и др. - отреагировали на это, создав организацию Open Software Foundation (OSF) для того, чтобы не зависеть от диктата одной единственной компании-поставщика операционных систем. OSF разработала UNIX-совместимую операционную систему, а также другие продукты без лицензионных ограничений со стороны одной компании. OSF/1 является модульной операционной системой, основанной на Mach, машинно-независимом мультипроцессорном ядре, разработанном в Carnegie-Mellon University (г. Питтсбург, США) в качестве инструмента для эмуляции других операционных систем. На основе Mach действительно удается одновременно эксплуатировать различные операционные системы на одной ЭВМ.

Для обеспечения переносимости OSF/1 совместима с AT&T UNIX System V и спецификациями программных интерфейсов Berkeley. Поскольку Mach и OSF/1 не содержит какого-либо кода UNIX, проблема лицензирования со стороны третьих компаний полностью снята.

В дополнение к средствам UNIX OSF/1 предлагает собственный набор функций, облегчающих разработку и выполнение программ. OSF/1 предназначена для работы в сетевой среде и поддерживает протокол TCP/IP. Файловая система OSF/1 также совместима со службой NFS протокола TCP/IP.

OSF разработала и другие продукты для распределенной вычислительной среды. OSF/Motif является графическим интерфейсом пользователя, обеспечивающим стандартное взаимодействие приложения с графическим терминалом.

Распределенная вычислительная среда (Distributed Computing Environment - DCE) представляет собой набор служб и средств для разработки, исполнения и поддержки приложений в распределенной среде. DCE может быть интегрирована с OSF/1, но является независимой от нее и в действительности может эксплуатироваться на базе других операционных систем.

Операционная система VAX/VMS E

VMS является операционной системой для ЭВМ компании Digital Equipment с 32-разрядным процессором серии VAX. VMS может применяться как в среде реального времени, так и в многопользовательской среде с соответствующими средствами защиты.

VMS предоставляет широкий набор функций, стандартный и ясный интерфейс для прямых обращений из программ. Это позволяет осуществлять интеграцию любых языков со всеми функциями операционной системы. Из функций реального времени VMS имеет почтовые ящики в форме логических, состоящих из записей файлов, возможность создания резидентных подпрограмм и обработку прерываний. Процесс в VMS может управлять условиями своего собственного исполнения (приоритет, распределение памяти), создавать другие процессы и управлять их исполнением. Иерархическое управление препятствует процессам с низким приоритетом модифицировать параметры исполнения процессов с высоким приоритетом.

В VMS возникают проблемы в случаях, когда предъявляются жесткие требования по времени, поэтому была разработана специальная версия, приспособленная для приложений РВ, которая называется VAX/ELN, состоящая из рабочей среды для исполнения прикладных программ на целевой ЭВМ и пакета для разработки программ с компиляторами для различных языков. Таким образом, операционная система предоставляет процессам логическую среду, состоящую из времени ЦП и оперативной памяти. ОС для многопользовательских приложений и приложений РВ имеют много общего, но приложения РВ могут требовать времени реакции порядка 1 мс. При программировании в РВ используются специальные функции для координации работы различных процессов. Для обычных программ этого не требуется. Кроме этого, программы РВ управляются прерываниями и могут явно ссылаться на время.

Центральная проблема многозадачного программирования и программирования в РВ - координация доступа к защищенным ресурсам. Стратегия разделения ресурсов, которая может основываться на простом циклическом или сложном динамическом механизме планирования, должна позволять избегать тупиков и блокировок, обеспечивать выделение ресурсов всем запрашивающим объектам и максимальную эффективность исполнения процессов. На нижнем уровне наиболее простым средством синхронизации является инструкция test_and_set. Наиболее часто используемые методы синхронизации и связи — это семафоры и почтовые ящики, которые в разных ОС реализуются по-разному. Результаты теории параллельного программирования играют важную роль на практике, так как соответствующие решения подкреплены формальными доказательствами. Это справедливо в особенности для систем РВ, поскольку тестирование программ представляет особую трудность. Применение проверенных

2. Операционные системы РВ OS-9 и VxWorks

1. Операционная система реального времени OS-9.
2. Операционная система VxWorks.

Операционная система РВ OS-9

OS-9 широко используются в системах автоматизации производства и телекоммуникационных системах, реализованных на базе микропроцессоров и микроконтроллеров фирмы **Motorola**.

Эта **ОСРВ** имеет две версии: **OS-9** написана на языке Ассемблера **Motorola 68K** и предназначена для работы с семействами **M680x0** и **M683xx**, **OS-9000** написана на языке **C** и может работать с семействами **MPC6xx**, **MPC5xx**, **MPCSxx**, **MCF52xx**, а также с микропроцессорами ряда других производителей: **Intel 486**, **Pentium**, **SPARC**, **MIPS**. Обе версии обеспечивают полную совместимость объектных кодов, поэтому для них обычно используется общее название **OS-9**. В качестве инструментального компьютера **OS-9** использует **IBM-PC**, работающие в среде **Windows**, или рабочие станции **SUN**, **HP**, **IBM RS/6000** с ОС типа **UNIX**.

Характерными особенностями **OS-9** являются модульность и гибкость ее структуры. Модульность обеспечивает возможность конфигурации целевой **ОСРВ** в соответствии с классом решаемых задач. За счет исключения неиспользуемых модулей достигается сокращение объема памяти и стоимости системы. Гибкость структуры позволяет достаточно просто и быстро производить реконфигурацию системы, расширение ее функциональных возможностей.

Все функциональные компоненты **OS-9** - ядро **РВ**, файловые менеджеры, средства (**OS-9 kernel**), средства общего управления внешними устройствами (**I/O man**), разработки - реализованы в виде автономных модулей. Комбинируя эти модули, разработчик может создавать целевые **ОС** различной конфигурации и функциональных возможностей - от несложных резидентных **ОСРВ**, хранящихся во внутреннем **ПЗУ** микроконтроллера, до сложно-функциональных многопользовательских систем разработки. Все модули **OS-9** могут размещаться в **ПЗУ**. Любые модули могут удаляться или добавляться с помощью простых команд, не требующих повторной компиляции или перекомпоновки. **OS-9** предоставляет пользователю возможность выбора ядра в зависимости от функционального назначения системы.

Ядро Atomic имеет малый объем (28 Кбайт) и обеспечивает минимальное время отклика. Например, при использовании микропроцессора **MC68040** с тактовой частотой **25 МГц** время реакции ядра на запрос прерывания составляет всего **3** мкс, что соответствует быстродействию систем "жесткого" **РВ**. Это ядро реализует минимальное число сервисных функций и применяется в системах, встраиваемых в различную аппаратуру. **Ядро Standard** обеспечивает выполнение широкого набора функций сервиса и разработки прикладных программ, но требует большого объема памяти - **67 Кбайт ПЗУ** и **38 Кбайт ОЗУ** для систем на базе **M68x0x**,**M683xx** (версия **OS-9**), до **512 Кбайт** для систем, использующих пакет поддержки обмена по сети **Интернет**, **75 Кбайт ПЗУ** и **24 Кбайт ОЗУ** для систем на базе **PowerPC** (версия **OS-9000**). Применение ядра **Standard** с различным набором других функциональных модулей позволяет реализовать системы различной сложности и назначения - от встраиваемых в аппаратуру контроллеров с резидентным программным обеспечением и простейшими средствами ввода-вывода до сложно-функциональных систем класса рабочих станций с

Модульная структура ОС РВ OS-9

В состав OS-9 входят более 20 файловых менеджеров, которые можно разделить на три группы: стандартные, сетевые и коммуникационные, графические и мультимедиа. Файловыми менеджерами, каждый из которых имеет определенное функциональное назначение и спецификацию, называются модули, управляющие логическими потоками данных.

Стандартные менеджеры входят в основной комплект системы и предназначены для выполнения базовых функций обмена с внешними устройствами. К ним относятся: pipeman - организующий очередь поступающих команд; ioman - выполняющий общее управление внешними устройствами; scf - управляющий байтовым последовательным обменом (связь с терминалом и другими устройствами по последовательному каналу); rbf - управляющий блочным обменом с прямым доступом памяти (связь с дисковой памятью); sbf - управляющий блочным последовательным обменом (связь с накопителями на магнитных лентах и другими устройствами); pcf - поддерживающий файловую DOS-систему (поставляется по требованию заказчика).

Сетевые и коммуникационные менеджеры обеспечивают работу OS-9 с различными сетями и обмен данными по каналам связи с наиболее распространенными стандартами протоколе обмена. Чаще всего из этой группы используются следующие менеджеры: nfs, nfm, isp - обеспечивающие основные протоколы сетевого обмена; profiman - реализующий протокол обмена с шиной Profibus; ism - реализующий обмен по сети цифровой связи стандарта ISDN; spf - обеспечивающий пакетный обмен по стандартному протоколу X.25; rtnfm - поддерживающий обмен по сети реального времени.

Для реализации графического интерфейса и работы с мультимедиа-приложениями используются файловые менеджеры: gfm - графический менеджер реального времени; mpfm - управляющий движущимися изображения; mfm - обеспечивающий пользовательский интерфейс с мультимедиа; g-windows - система оконного графического интерфейса, разработанная фирмой Gespac в виде файлового менеджера для OS-9.

Физический интерфейс ОС РВ OS-9

Физический интерфейс OS-9 с разнообразными внешними устройствами обеспечивается большим набором драйверов, которые созданы как фирмой Microware Systems, так и многочисленными разработчиками аппаратуры, использующей эту операционную систему для конкретных приложений.

Большинство драйверов, реализующих интерфейс со стандартными внешними устройствами, входят в пакет, из которого пользователь может выбрать необходимые средства для своего проекта.

В составе OS-9 имеется также пакет программ BSP для поддержки плат развития, который обеспечивает совместную работу OS-9 с целым рядом SBC, включая SBC типа MVME162, 172, 1603, 1604 фирмы Motorola. Используя BSP, OS-9 и какой-либо из этих SBC, разработчик может быстро сконфигурировать мощную целевую систему для своего конкретного приложения.

OS-9 содержит средства поддержки программирования, позволяющие проектировщику создавать прикладное программное обеспечение. Эти средства включают компиляторы Ultra C/C++, текстовый редактор EMACS, три вида отладчиков, в том числе символьные, а также разнообразные утилиты для организации контроля и сборки программных проектов. Кроме того, проектировщик может использовать большой набор совместимых с OS-9 текстовых редакторов, компиляторов C/C++, Forth, Ada, Modula-2 и других языков, которые разработаны рядом других фирм.

Для удобства пользователя совместно с OS-9 поставляются набор средств программирования-отладки OS-9 Tool Kit, интегрированная среда разработки FasTrak. В состав OS-9 Tool Kit входят основные средства разработки программ, указанные выше.

Интегрированная среда разработки FasTrak

Интегрированная среда разработки FasTrak предоставляет пользователю наиболее полный комплект средств программирования-отладки. FasTrak имеет две версии: для функционирования в среде Windows на инструментальных компьютерах IBM-PC; для функционирования с системой UNIX на рабочих станциях SUN, HP, IBM RS/6000.

Часть программных средств FasTrak инсталлируется на инструментальном компьютере, а часть - на целевой системе пользователя. Интерфейс инструментального компьютера и целевой системы осуществляется файловым менеджером isp, который реализует протокол TCP/IP, обеспечивая связь по последовательному каналу или по сети Ethernet.

Среда FasTrak интегрирует все средства, необходимые для поддержки проектирования-отладки целевых систем. Версия FasTrak для IBM-PC содержит высокоэффективный текстовый редактор Premia's Codewright, который имеет средства перекодировки клавиатуры, обеспечивающие пользователю возможность вести редактирование в удобном для него формате. Версия для UNIX-станций позволяет использовать любой редактор, функционирующий с ОС UNIX. В состав FasTrak входят компиляторы Ultra C/C++, возможно также использование других компиляторов, например GNU C/C++ фирмы Cygnus Support. Отладчики FasTrak обеспечивают два режима отладки: пользовательский для создания прикладных программ, и системный, который выполняет обслуживание прерываний, системных вызовов и обращение к ядру PB. Реализуется также отладка мультипроцессорных систем. При выполнении контрольных прогонов рабочей программы программа-профилировщик дает информацию о количестве обращений к различным программным модулям и времени их выполнения. В составе среды FasTrak имеются средства интерфейса с логическими анализаторами фирмы Hewlett-Packard и схемными эмуляторами фирм Hewlett-Packard, EST, Applied Microsystems, Orion. Широкий набор функциональных возможностей делает среду FasTrak эффективным средством создания программного обеспечения для микропроцессорных и микроконтроллерных систем. Модульная структура ОСРВ OS-9 позволяет легко конфигурировать ее в соответствии с потребностями заказчиков.

В настоящее время фирма Microware Systems поставляет ряд системных пакетов, ориентированных на различные сферы приложения: Wireless OS-9 - для разработки устройств беспроводной связи: сотовых телефонов, пейджеров, портативных цифровых ассистентов (PDA); Internet OS-9 - для разработки устройств с доступом к сети Интернет; Digital Audio/Video Interactive Decoder (DAVID) OS-9 - для разработки распределенных систем цифрового интерактивного телевидения.

Таким образом, ОСРВ OS-9 позволяет удовлетворить запросы широкого круга разработчиков, создающих

Операционная система VxWorks

VxWorks относится к классу систем "жесткого" РВ. Эта ОСРВ предназначена для работы с семействами M680xO, M683xx, MPC6xx, MPC5xx, MPCSxx, MCF52xx, а также с микропроцессорами других производителей: Intel 486, Pentium, SPARC, MIPS, DEC Alpha, HP PA-RISC.

В качестве инструментального компьютера используются IBM-PC, работающие в среде Windows, или рабочие станции SUN, HP, DEC, IBM RS/6000 с операционными системами типа UNIX. При выполнении отладки VxWorks, которая инсталлируется на отлаживаемой целевой системе, работает совместно с интегрированной средой разработки Tornado, функционирующей на инструментальном компьютере .

VxWorks имеет иерархическую организацию, нижним уровнем которой служит микроядро РВ, выполняющее базовые функции планирования задач и управления их коммуникацией-синхронизацией. Все остальные функции - управление памятью, вводом-выводом, сетевым обменом и другие, реализуются дополнительными модулями. Микроядро с минимальным набором модулей занимает 20...40 Кбайт памяти. Для встроенных систем, имеющих жесткие ограничения на объем памяти, разработано редуцированное ядро Wind Stream, которое требует для работы всего 8 Кбайт ПЗУ и 2 Кбайт ОЗУ.

Для реализации графических приложений используется система графического интерфейса VX-Windows. В тех случаях, когда ограниченный объем памяти целевой системы не позволяет использовать VX-Windows, предлагается графическая библиотека RTGL, которая содержит базовые графические примитивы, наборы шрифтов и цветов, драйверы типовых устройств ввода и графических контроллеров. В состав VxWorks входят также различные средства поддержки разнообразных сетевых протоколов: X.25, ISDN, ATM, SS7, Frame Relay и ряда других.

Специальные средства отладки в реальном масштабе времени обеспечивают трассировку заданных событий и их накопление в буферной памяти для последующего анализа. Трассировку системных событий выполняет динамический анализатор WindView, который работает аналогично логическому анализатору, отображая на экране временные диаграммы переключения задач, записи в очередь сообщений, установки программных светофоров и другие процессы. Монитор данных StethoScope позволяет анализировать динамическое изменение пользовательских и системных переменных, включая в себя также профилировщик процедур.

В составе VxWorks имеется пакет программ BSP для постановки данной ОСРВ на ряд плат развития, включая SBC фирмы Motorola, что позволяет конфигурировать таким образом целевую систему для конкретного

Операционная система VxWorks

Симулятор VxSim позволяет моделировать на инструментальном компьютере многозадачную среду VxWorks и интерфейс с целевой системой. Он позволяет разрабатывать и отлаживать программное обеспечение без подключения целевой системы.

Среда VxWorks обеспечивает также возможности программирования мультипроцессорных систем.

Для поддержки программирования предлагается интегрированная среда разработки Tornado, в состав которой входит VxWorks 5.3 - ядро PB и системные библиотеки, средства программирования C/C++ Toolkit, высокоуровневый отладчик CrossWind и ряд других средств. Пакет C/C++ Toolkit содержит компиляторы GNU C/C++ фирмы Cygnus Support. Отладчик CrossWind является расширенной версией отладчика GDB фирмы Cygnus Support. Он имеет графический пользовательский интерфейс и поддерживает отладку как на прикладном, так и на системном уровне. Дополнительные средства среды Tornado обеспечивают управление процессом отладки, визуализацию состояния целевой системы, другие сервисные функции.

Tornado может использоваться совместно с VX-Windows, WindView, StethoScope, VxSim и рядом других средств из состава VxWorks.

Характерной особенностью среды Tornado является ее открытая архитектура, которая позволяет пользователю подключать собственные специализированные инструментальные средства и расширять возможности стандартных. Открытость реализована с помощью прикладных программных интерфейсов API, которые дают возможность различным программным продуктам обмениваться между собой данными на инструментальном компьютере и взаимодействовать с VxWorks, установленной на целевой системе.

ОСРВ VxWorks вместе с интегрированной средой Tornado является мощным средством реализации целевых систем, работающих в условиях жестких ограничений на объем используемой памяти и время отклика на внешние события.

Сетевая операционная система реального времени QNX

1. Принципы построения СРВ QNX.
2. Архитектура системы QNX.
3. Основные механизмы QNX для организации распределенных вычислений.

Принципы построения СРВ QNX

Операционная система QNX является мощной операционной системой, позволяющей проектировать сложные программные системы, работающие в реальном времени как на одном компьютере, так и в локальной вычислительной сети.

Встроенные средства ОС QNX обеспечивают поддержку многозадачного режима на одном компьютере и взаимодействие параллельно выполняемых задач на разных компьютерах, работающих в среде локальной вычислительной сети. Основной язык программирования в системе - С. Основная операционная среда соответствует стандартам POSIX-интерфейса. Это позволяет с небольшими доработками перенести необходимое накопленное программное обеспечение в среду ОС QNX для организации их работы в среде распределенной обработки. ОС QNX является сетевой, мультизадачной, многопользовательской (многотерминальной) и масштабируемой. С точки зрения пользовательского интерфейса и API она похожа на UNIX. QNX была разработана канадской фирмой QNX Software Systems Limited в 1989 году по заказу Минобороны США. Причем эта система построена на архитектурных принципах, отличных от принципов, использованных при создании ОС UNIX.

QNX была первой коммерческой ОС, построенной на принципах микроядра и обмена сообщениями. Система реализована в виде совокупности независимых 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. QNX является операционной системой, которая дает полную гарантию в том, что процесс с наивысшим приоритетом начнет выполняться практически немедленно, и что критическое событие (например, сигнал тревоги) никогда не будет потеряно.

Масштабируемость и эффективность, достигаемые оптимальным использованием ресурсов и означающие ее применимость для встроенных систем. Драйверы и менеджеры можно запускать и удалять (кроме файловой системы) динамически, из командной строки.

Расширяемость и надежность одновременно, поскольку написанный вами драйвер не нужно компилировать в ядро. Менеджеры ресурсов (сервис логического уровня) работают в кольце защиты 3, и возможно добавлять свои, не опасаясь за систему. Драйверы работают в кольце с уровнем привилегий 1 и могут вызвать проблемы, но не фатального характера.

Быстрый сетевой протокол FLEET, прозрачный для обмена сообщениями, автоматически обеспечивающий отказоустойчивость, балансирование нагрузки и маршрутизацию между альтернативными путями доступа.

Архитектура CPB QNX

Первым обязательным требованием к архитектуре ОСРВ является многозадачность. Очевидно, что варианты с псевдомногозадачностью (а точнее - не вытесняющая многозадачность) типа MS Windows 3.x или Novell NetWare неприемлемы, поскольку они допускают возможность блокировки или даже полного развала системы одним неправильно работающим процессом. Для предотвращения блокировок ОСРВ должна использовать квантование времени (то есть вытесняющую многозадачность).

Вторая проблема (организация надежных вычислений) может быть эффективно решена при полном использовании возможностей процессоров Intel 80386 и старше, что предполагает работу ОС в 32-разрядном режиме процессора. Для эффективного обслуживания прерываний ОС должна использовать алгоритм диспетчеризации, обеспечивающий вытесняющее планирование, основанное на приоритетах. Крайне желательны эффективная поддержка сетевых коммуникаций и наличие развитых механизмов взаимодействия между процессами, поскольку реальные технологические системы обычно управляются целым комплексом компьютеров и/или контроллеров. Важно, чтобы ОС поддерживала множественные потоки управления (не только мультипрограммный, но и мультизадачный режимы), а также симметричную мультипроцессорность.

При соблюдении этих условий, ОС должна быть способна работать на ограниченных аппаратных ресурсах, поскольку одна из ее основных областей применения - это встроенные системы. К сожалению, данное условие обычно реализуется путем урезания стандартных сервисных средств.

QNX - это ОС реального времени, позволяющая эффективно организовать распределенные вычисления. В системе реализована концепция связи между задачами на основе сообщений, посылаемых от одной задачи к другой, причем задачи эти могут находиться как на одном и том же компьютере, так и на удаленных, но связанных локальной вычислительной сетью. Реальное время и концепция связи между процессами в виде сообщений оказывают решающее влияние на разрабатываемое для QNX программное обеспечение и на программиста, стремящегося с максимальной выгодой использовать преимущества системы.

Микроядро имеет объем и несколько десятков килобайт (в одной из версий - 10 Кбайт, в другой - менее 32 Кбайт), то есть это одно из самых маленьких ядер среди всех существующих операционных систем. В этом объеме помещаются:

. механизм передачи сообщений между процессами (IPC);

. редиректор прерываний;

. блок планирования выполнением задач;

Архитектура СРВ QNX

Механизм передачи межпроцессорных сообщений занимается пересылкой сообщений между процессами и является одной из важнейших частей операционной системы, так как все общения между процессами, в том числе и системными, происходит через сообщения.

Сообщение в QNX - это последовательность байтов произвольной длины (0-65 535 байтов) произвольного формата. Протокол обмена сообщениями выглядит таким образом. Например, задача блокируется для ожидания сообщения. Другая же задача посылает первой сообщение и при этом блокируется сама, ожидая ответа. Первая задача разблокируется, обрабатывает сообщение и отвечает, разблокируя при этом вторую задачу.

Сообщения и ответы, пересылаемые между процессами при их взаимодействии, находятся в теле отправляющего их процесса до того момента, когда они могут быть приняты. Это значит, что, с одной стороны, уменьшается вероятность повреждения сообщения в процессе передачи, а с другой - уменьшается объем оперативной памяти, необходимый для работы ядра. Кроме того, уменьшается число пересылок из памяти в память, что разгружает процессор. Особенностью процесса передачи сообщений является то, что в сети, состоящей из нескольких компьютеров под управлением QNX, сообщения могут прозрачно передаваться процессам, выполняющимся на любом из узлов. Определены в QNX еще и два дополнительных метода передачи сообщений - метод представителей (Proxy) и метод сигналов (Signal).

Представители используются в случаях, когда процесс должен передать сообщение, но не должен при этом блокироваться на передачу. В таком случае вызывается функция qnx_proxy_attach( ) и создается представитель. Он накапливает в себе сообщения, которые должны быть доставлены другим процессам. Любой процесс, знающий идентификатор представителя, может вызвать функцию Trigger( ), после чего будет доставлено первое в очереди сообщение. Функция Trigger( ) может вызываться несколько раз, и каждый раз представитель будет доставлять следующее сообщение. При этом представитель содержит буфер, в котором может храниться до 65 535 сообщений.

Сигналы, поддерживаемые СРВ QNX

Система QNX поддерживает множество сигналов, совместимых с POSIX, большое количество сигналов, традиционно использовавшихся в UNIX (поддержка этих сигналов реализована для совместимости с переносимыми приложениями, и ни один из системных процессов QNX их не генерирует), а также несколько сигналов, специфичных для самой QNX.

По умолчанию любой сигнал, полученный процессом, приводит к завершению процесса (кроме нескольких сигналов, которые по умолчанию игнорируются). Но процесс с приоритетом уровня «superuser» может защититься от нежелательных сигналов. В любом случае процесс может содержать обработчик для каждого возможного сигнала. Сигналы удобно рассматривать как разновидность программных прерываний.

Редиректор прерываний является частью ядра и занимается перенаправлением аппаратных прерываний в связанные с ними процессы. Благодаря такому подходу возникает один побочный эффект - с аппаратной частью компьютера работает ядро, оно перенаправляет прерывания процессам - обработчикам прерываний. Обработчики прерываний обычно встроены в процессы, хотя каждый из них исполняется асинхронно с процессом, в который он встроен. Обработчик исполнялся в контексте процесса, и имеет доступ ко всем глобальным переменным процесса. При работе обработчика прерываний прерывания разрешены, но обработчик приостанавливается только в том случае, если произошло более высокоприоритетное прерывание. Если это позволяется аппаратной частью, к одному прерыванию может быть подключено несколько обработчиков, и каждый из них получит управление при возникновении прерывания.

Этот механизм позволяет пользователю избежать работы с аппаратным обеспечением напрямую и тем самым избегать конфликтов между различными процессами, работающими с одним и тем же устройством. Для обработки сигналов от внешних устройств, чрезвычайно важно минимизировать время между возникновением события и началом непосредственной его обработки. Этот фактор является существенным в любой области применения, от работы терминальных устройств до обработки высокочастотных сигналов.

Блок планирования выполнения задач (диспетчер задач) занимается обеспечением многозадачности. В этой части ОС QNX предоставляет разработчику огромный простор для выбора той методики выделения ресурсов процессора задаче, которая обеспечит наиболее подходящие условия для критических приложений или обеспечит такие условия для некритических приложений, что они выполнятся за разумное время, не мешая работе критических приложений.

К выполнению своих функций как диспетчера ядро приступает в следующих случаях:

. какой-либо процесс вышел из блокированного состояния;

. истек квант времени для процесса, владеющего CPU;

. работающий процесс прерван каким-либо событием.

Сигналы, поддерживаемые СРВ QNX

В QNX существуют три метода диспетчеризации: FIFO (первым пришел - первым обслужен), Round-Robin (процессу выделяется определенный квант времени для работы) и адаптивный, который является наиболее используемым.

Первый наиболее близок к кооперативной многозадачности. То есть процесс выполняется до тех пор, пока он не перейдет в состояние ожидания сообщения, состояние ожидания ответа на сообщение или не отдаст управление ядру. При переходе в одно из таких состояний процесс помещается последним в очередь процессов с таким же уровнем приоритета, а управление передается процессу с наибольшим приоритетом.

Во втором варианте все происходит так же, как и в предыдущем, с той разницей, что период, в течение которого процесс может работать без перерыва, ограничивается неким квантом времени.

Процесс, работающий с адаптивным методом, в QNX ведет себя следующим образом: 1. Когда процесс полностью использовал выделенный ему квант времени, его приоритет снижается на 1, если в системе есть процессы с тем же уровнем приоритета, готовые к исполнению. 2. Если процесс с пониженным приоритетом остается не обслуженным в течение секунды, его приоритет увеличивается на 1. Если процесс блокируется, ему возвращается оригинальное значение приоритета.

По умолчанию процессы запускаются в режиме адаптивной многозадачности. В этом же режиме работают все системные утилиты QNX. Процессы, работающие в разных режимах многозадачности, могут одновременно находиться в памяти и исполняться. Важный элемент реализации многозадачности — это приоритет процесса. Обычно приоритет процесса устанавливается при его запуске. Но есть дополнительная возможность, называемая «вызываемый клиентом приоритет». Как правило, она реализуется для серверных процессов. При этом приоритет процесса-сервера устанавливается только на время обработки запроса и становится равным приоритету процесса-клиента.

Сетевой интерфейс в системе QNX является неотъемлемой частью ядра. Он взаимодействует с сетевым адаптером через сетевой драйвер, но базовые сетевые сервисы реализованы на уровне ядра. Благодаря такой организации сеть превращается в однородную вычислительную среду. При этом для большинства приложений не имеет значения, с какого компьютера они были запущены, на каком исполняются и куда поступают результаты их работы. Такое решение принципиально отличает QNX от остальных ОС, которые тоже имеют все необходимые средства для работы в сети, и делает системы, работающие под ее управлением, по-настоящему распределенными. Все сервисы QNX, не реализованные непосредственно в ядре, работают как стандартные процессы в полном соответствии с основными концепциями микроядерной архитектуры. С точки зрения операционной системы системные процессы ничем не отличаются от всех

Основные механизмы QNX для организации распределенных вычислений

QNX является сетевой ОС и позволяет организовать эффективные распределенные вычисления. Для организации сети на каждой машине, называемой узлом, помимо ядра и менеджера процессов должен быть запущен менеджер Net, который не зависит от аппаратной реализации сети.

Данная аппаратная независимость обеспечивается за счет использования сетевых драйверов. В QNX имеются драйверы для различных сетей, например Ethernet, Arcnet, Token Ring. Кроме этого, имеется возможность организации сети через последовательный канал или модем.

В QNX четвертой версии полностью реализовано встроенное сетевое взаимодействие «точка-точка». Любые ресурсы (модемы, диски, принтеры) могут быть добавлены к системе простым их подключением к любой машине в сети. QNX поддерживает одновременную работу в сетях Ethernet, Arcnet, Serial и Token Ring, обеспечивает более чем единственный путь для коммуникации, а также балансировку нагрузки в сетях. Если кабель или сетевая плата выходит из строя таким образом, что связь через эту сеть прекращается, то система будет автоматически перенаправлять данные через другую сеть. Это происходит в режиме «online», предоставляя пользователю автоматическую сетевую избыточность и увеличивая скорость коммуникаций во всей системе.

Каждому узлу в сети соответствует уникальный целочисленный идентификатор - логический номер узла. Любой поток в сети QNX имеет прозрачный доступ ко всем ресурсам сети, то же самое относится и к взаимодействию потоков. Для взаимодействия потоков, находящихся на разных узлах сети, используются те же самые вызовы ядра, что и для потоков, выполняемых на одном узле. В том случае, если потоки находятся на разных узлах сети, ядро переадресует запрос менеджеру сети. Для организации обмена в сети используется надежный и эффективный протокол транспортного уровня FLEET. Каждый из узлов может принадлежать одновременно нескольким QNX-сетям. Если сетевое взаимодействие может быть реализовано несколькими путями, для передачи выбирается незагруженная и более скоростная сеть.

Сетевое взаимодействие является узким местом в большинстве ОС и обычно создает значительные проблемы для СРВ. Для решения проблемы, разработчики QNX создали собственную специальную сетевую технологию FLEET и соответствующий протокол транспортного уровня FTL (FLEET transport layer), который является уникальным. Благодаря этой технологии сеть компьютеров с QNX фактически можно представлять как один виртуальный суперкомпьютер. Все ресурсы любого из узлов сети автоматически доступны другим. Например, утилита make в QNX автоматически распараллеливает компиляцию пакетов из нескольких модулей на все доступные узлы сети, а затем собирает исполняемый модуль по мере завершений компиляции на узлах. Специальный драйвер, входящий в комплект поставки, позволяет использовать для сетевого взаимодействия любое устройство, с которым может быть ассоциирован

Принципы построения СРВ QNX

Достигаются все эти удобства за счет того, что поддержка сети частично обеспечивается и микроядром (специальный код в его составе позволяет QNX фактически объединять все микроядра в сети в одно ядро).

Fault-Tolerant Networking	QNX может одновременно использовать несколько физических сетей. При выходе из строя любой из них данные будут автоматически перенаправлены «на лету» через другую сеть
Load-Balancing on the Fly	При наличии нескольких физически соединений QNX автоматически распараллеливает передачу пакетов по соответствующим сетям
Efficient. Performance	Специальные драйверы, разрабатываемые фирмой QSSL для широкого спектра оборудования, позволяют с максимальной эффективностью использовать сетевое оборудование
Extensable Architecture	Любые новые типы сетей могут быть поддержаны путем добавления соответствующих драйверов
Transparent Distributed Processing	Благодаря отсутствию разницы между передачей сообщений в пределах одного узла и между узлами нет необходимости вносить какие-либо изменения в приложении для того, чтобы они могли взаимодействовать через сеть

Когда ядро получает запрос на передачу данных процессу, находящемуся на удаленном узле, оно переадресовывает этот запрос менеджеру Net, в подчинении которого находятся драйверы всех сетевых карт. Менеджер Net может отслеживать состояние каждой сети и динамически перераспределять нагрузку между ними. В случае, когда одна из сетей выходит из строя, информационный поток автоматически перенаправляется в другую доступную сеть, что важно при построении высоконадежных систем. Кроме поддержки своего собственного протокола, Net обеспечивает передачу пакетов TCP/IP, SMB и многих других, используя то же сетевое оборудование. Производительность QNX в сети приближается к производительности аппаратного обеспечения.

При проектировании системы реального времени, как правило, необходимо обеспечить одновременное выполнение нескольких приложений. В QNX/Neutrino параллельность выполнения достигается за счет использования потоковой модели POSIX, в которой процессы в системе представляются в виде совокупности потоков. Поток является минимальной единицей выполнения и диспетчеризации для ядра Neutrino, процесс определяет адресное пространство для потоков. Каждый процесс состоит минимум из одного потока. QNX предоставляет богатый набор функций для синхронизации потоков. В отличие от потоков само ядро не подлежит диспетчеризации. Код ядра исполняется только в том случае, когда какой-нибудь поток вызывает функцию ядра или при обработке аппаратного прерывания.

Концепция передачи сообщений в QNX

QNX базируется на концепции передачи сообщений. Передачу сообщений, а также их диспетчеризацию осуществляет ядро системы. Кроме того, ядро управляет временными прерываниями. Выполнение остальных функций обеспечивается задачами-администраторами.

Программа, желающая создать задачу, посылает сообщение администратору задач (модуль task) и блокируется для ожидания ответа. Если новая задача должна выполняться одновременно с порождающей ее задачей, администратор задач task создает ее и, отвечая, выдает порождающей задаче идентификатор (ID) созданной задачи. В противном случае никакого сообщения не посылается до тех пор, пока новая задача не закончится сама по себе. В этом случае в ответе администратора задач будут содержаться конечные характеристики закончившейся задачи. Сообщения отличаются количеством данных, передающихся от одной задачи к другой. Данные копируются из адресного пространства первой задачи в адресное пространство второй, и выполнение первой задачи приостанавливается до тех пор, пока вторая не вернет ответное сообщение. В действительности обе задачи кратковременно взаимодействуют во время выполнения передачи. При передаче сообщения имеет значение только длина сообщения (максимум 65 535 байт). Для сообщений могут использоваться несколько протоколов.

Основные операции над сообщениями: послать, получить и ответить, а также несколько их вариантов для обработки специальных ситуаций. Получатель всегда определяется своим идентификатором задачи, хотя существуют способы ассоциировать имена с идентификатором задачи. Наиболее интересные варианты операций включают в себя возможность получать (копировать) только первую часть сообщения, а оставшуюся часть отдельными частями. Это можно использовать для того, чтобы сначала узнать длину сообщения, а затем динамически распределить принимающий буфер. Если необходимо задержать ответное сообщение до получения и обработки другого сообщения, то чтение первых нескольких байт дает компактный «обработчик», через который позже можно получить доступ ко всему сообщению. Таким образом, задача предохраняется от необходимости хранить в себе большое количество буферов.

Другие функции позволяют программе получать сообщения только тогда, когда она уже ожидает их приема, а не блокироваться до получения сообщения, а также его трансляции другой задаче без изменения идентификатора передатчика. Задача, которая транслировала сообщение, в транзакции невидима. Кроме этого, QNX обеспечивает объединение сообщений в структуру данных, называемую очередью. Очередь - это область данных в третьей, отдельной задаче, которая временно принимает передаваемое сообщение и немедленно отвечает передатчику. В отличие от стандартной передачи сообщений, передатчик немедленно освобождается для продолжений своей работы. Задача администратора очереди хранить в себе сообщение до тех пор, пока приемник не будет готов прочитать его, что делается путем запроса сообщения у администратора очереди. Любое количество сообщений может храниться в очереди, причем

Порты в СРВ QNX

Помимо сообщений и очередей в QNX для взаимодействия задач и организации распределенных вычислений имеются так называемые порты, которые позволяют формировать сигнал одного конкретного условия, и механизм исключений.

Порт подобен флагу, известному всем задачам на одном и том же узле (но не на различных узлах). Он имеет два состояния, которые могут трактоваться как «присоединить» и «освободить», хотя пользователь может использовать свою интерпретацию; например, «занят» и «доступен». Порты используются для быстрой простой синхронизации между задачей и обработчиком прерываний устройства. Они нумеруются от нуля до максимума 32 (на некоторых типах узлов возможно и больше). Первые 20 номеров зарезервированы для использования операционной системой.

С портом могут быть выполнены три операции:

- присоединить порт;
- отсоединить порт;
- послать сигнал в порт.

Одновременно к порту может быть присоединена только одна задача Если другая задача попытается «отсоединиться» от того же самого порта, то произойдет отказ при вызове функции, и управление вернется к задаче, которая в настоящий момент присоединена к этому порту. Это самый быстрый способ обнаружить идентификатор другой задачи, подразумевая, что задачи могут договориться использовать один номер порта. Все рассматриваемые задачи должны находиться на одном и том же узле. При работе нескольких узлов специальные функции обеспечивают большую гибкость и эффективность.

Любая задача может посылать сигнал в любой порт независимо от того, была ли она присоединена к нему или нет (предпочтительно, чтобы не была присоединена). Сигнал подобен не блокирующей передаче пустого сообщения. То есть передатчик не приостанавливается, а приемник не получает какие-либо данные, он только отмечает, что конкретный порт изменил свое состояние.

Задача, присоединенная к порту, может ожидать прибытия сигнала или может периодически читать порт. QNX хранит информацию о сигналах, передаваемых в каждый порт, и уменьшает счетчик после каждой операции «приема» сигнала («чтение» возвращает счетчик и устанавливает его в ноль). Сигналы всегда принимаются перед сообщениями, давая им тем самым больший приоритет над сообщениями. В этом смысле сигналы часто используются обработчиками прерываний

Исключения СРВ QNX

В отличие от описанных выше методов, которые строго синхронизируются, «исключения» обеспечивают асинхронное взаимодействие. То есть исключение может прервать нормальное выполнение потока задачи. Они, таким образом, являются аварийными событиями.

QNX резервирует для себя 16 исключений для того, чтобы оповещать задачи о прерывании с клавиатуры, нарушении памяти и подобных необычных ситуациях. Остальные 16 исключений могут быть определены и использованы прикладными задачами.

Системная функция может быть вызвана для того, чтобы позволить задаче управлять своей собственной обработкой исключений, выполняя свою собственную внутреннюю функцию во время возникновения исключения.

Функция исключения задачи вызывается асинхронно операционной системой, а не самой задачей. Вследствие этого исключения могут иметь сильнодействующее побочное влияние на операции (например, передачу сообщений), которые выполняются в это же время. Обработчики исключений должны быть написаны очень аккуратно.

Одна задача может установить одно или несколько исключений на другой задачей. Они могут быть комбинацией системных исключений (определенных выше) и исключений, определяемых приложениями, что обеспечивает другие возможности для межзадачного взаимодействия.

Благодаря такому свойству QNX, как возможность обмена посланиями между задачами и узлами сети, программы не заботятся о конкретном размещении ресурсов в сети. Это свойство придает системе необычную гибкость. Так, узлы могут произвольно добавляться и изыматься из системы, не затрагивая системные программы. QNX приобретает эту конфигурационную независимость благодаря применению концепции о «виртуальных» задачах. У виртуальных задач непосредственный код и данные, будучи на одном из удаленных узлов, возникают и ведут себя так, как если бы они были локальными задачами какого-то узла со всеми атрибутами и привилегиями. Программа, посылающая сообщение в сети, никогда не посылает его точно. Сначала она открывает «виртуальную цепочку». Виртуальная цепочка включает все виртуальные задачи, связанные между собой. На обоих концах такой связи имеются буферы, которые позволяют хранить самое большое послание из тех, которые цепочка может нести в данном сеансе связи. Сетевой администратор помещает в эти буферы все сообщения для соединенных задач. Виртуальная задача, таким образом, занимает всего лишь пространство, необходимое для буфера и входа в таблицу задач. Чтобы открыть связь, необходимо знать идентификатор узла и задачи, с которой устанавливается связь. Для этого необходимо знать идентификатор задачи администратора, ответственного за данную функцию, или глобальное имя сервера. Не раскрывая здесь подробно механизм обмена

Контрольные вопросы:

1. Перечислите основные параметры операционных систем реального времени.
2. Дайте характеристику времени реакции системы на прерывание.
3. Поясните смысл параметра операционных систем реального времени «время переключения контекста».
4. Приведите примеры размера ядра операционных систем реального времени.
5. Дайте характеристику механизмам систем реального времени.
6. Что понимается под идеальной операционной системой реального времени?
7. Какие параметры указываются в каждом описателе операционных систем реального времени?
8. Какие алгоритмы планирования операционных систем Вам известны? Дайте их характеристику.
9. Дайте характеристику механизмам межзадачного взаимодействия операционных систем реального времени.
10. Какие базовые концепции операционных систем реального времени Вы знаете?
11. Дайте характеристику монолитной архитектуре операционных систем реального времени. Нарисуйте ее модель.
12. Перечислите основные достоинства и недостатки монолитной архитектуры.
13. Какие недостатки имеет ОСРВ модульной архитектуры на основе микроядра?
14. Как осуществляется взаимодействие между компонентами системы и пользовательскими процессами в объектной архитектуре на основе объектов-микроядер?
15. Дайте характеристику ОСРВ объектной архитектуры на основе объектов-микроядер.
16. Почему про QNX часто говорят «сетевая» ОС?
17. Что такое сетевой протокол FLEET? 10.Какие функции реализует ядро QNX?
18. В чем вы видите принципиальные отличия между ядром Windows NT 4.0, которое считают построенным по микроядерным принципам, от ядра QNX?

Thank You!