Федеральное агентство связи
Государственное образовательное учреждение высшего профессионального образования
«Поволжский государственный университет телекоммуникаций и информатики»

Кафедра «Программное обеспечение и управление в технических системах»
(наименование кафедры)

«УТВЕРЖДАЮ»
Заведующий кафедрой ПОУТС
наименование кафедры
_____ Тарасов В.Н.
подпись, Фамилия И.О.
« 31 » 08 2011 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Системы реального времени
(наименование учебной дисциплины)

Для специальности: 230105 – Программное обеспечение вычислительной техники и автоматизированных систем
(код и наименование направления (специальности) подготовки)

Обсуждено на заседании кафедры
ПОУТС

« 31 » 08 2011 г.

протокол № 1

Самара
2011

Федеральное агентство связи
Государственное образовательное учреждение высшего профессионального образования
«Поволжский государственный университет телекоммуникаций и информатики»

Кафедра «Программное обеспечение и управление в технических системах»
(наименование кафедры)

КОНСПЕКТ ЛЕКЦИЙ

ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Системы реального времени

(наименование учебной дисциплины)

по специальности :

230105 – Программное обеспечение вычислительной техники и автоматизированных систем
наименование специальности (направления подготовки)

Самара
2011

УДК 621.391

Сивков В.С.
Системы реального времени. Конспект лекций. – Самара.: ПГУТИ, 2011. – 21 с.

Дисциплина системы реального времени (СРВ) рассматривает общие вопросы архитектуры систем обработки данных, функционирующих в режиме реального времени. Рассматриваются вопросы связанные с механизмами реального времени, функциями операционных систем реального времени (ОСРВ), рассматриваются стандарты для ОСРВ. Рассматриваются общие вопросы построения и работы ОСРВ, а также проводится обзор различных реализаций ОСРВ.

Рецензент:
Ситникова С.В. – к.т.н., доцент, доцент кафедры «Электродинамики и антенн» ФГОБУ ВПО ПГУТИ

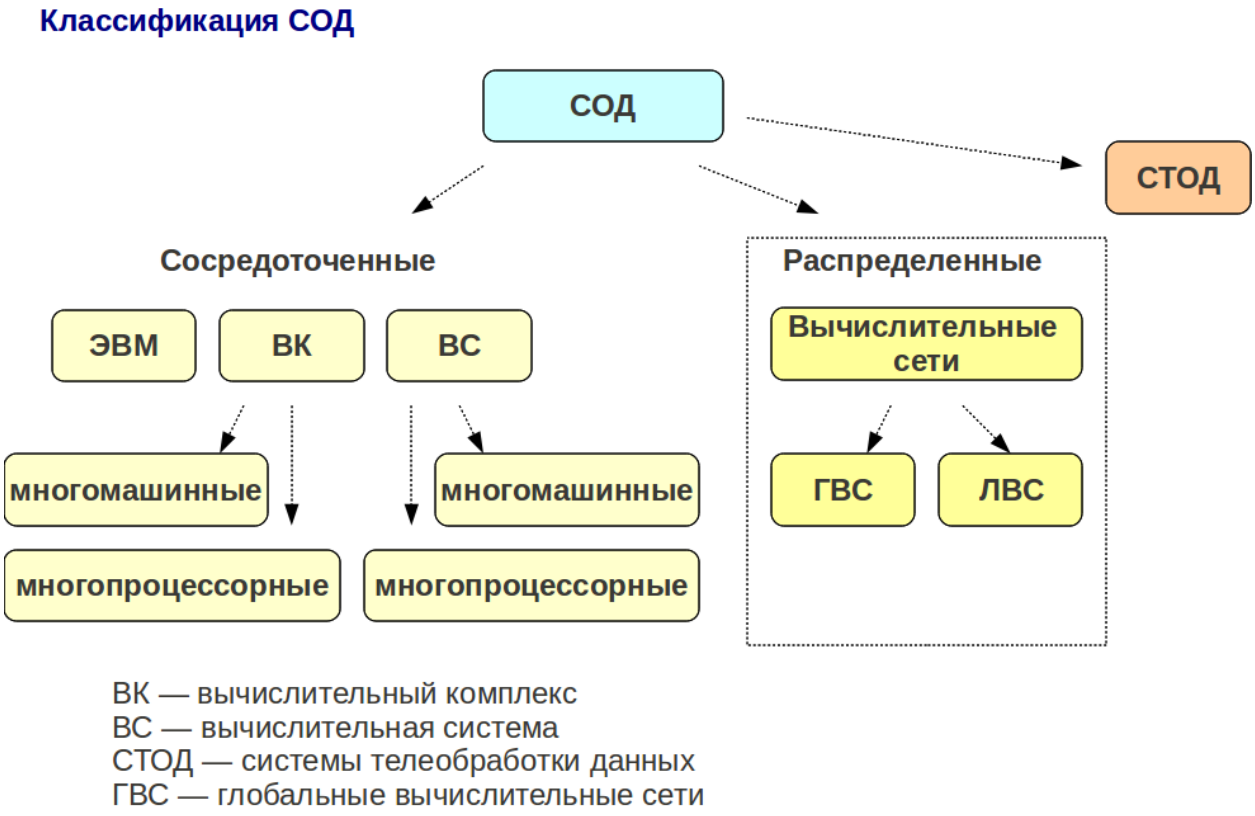
Государственное образовательное учреждение высшего профессионального образования
«Поволжский государственный университет телекоммуникаций и информатики»

Содержание конспекта лекций

Лекция 1

Системы обработки данных — СОД

СОД – это совокупность технических средств и программного обеспечения, предназначенных для информационного обслуживания пользователя и технических объектов.



Типы СОД

Назначение информационных систем – поиск и анализ информации, потребителем которой является человек. Основу алгоритмов работы такой системы составляют программы логической обработки данных. Допустимое время обработки данных определяется максимально возможным временем ожидания. В таких системах, как правило, объем входной информации невелик, но в них присутствует большие постоянные и медленно изменяющиеся массивы данных.

Назначение управляющих систем – целенаправленное изменение состояние объекта или управление процессом функционирования объекта. Для управления необходимо знать состояние объекта в заданный момент времени, каковы внешние воздействия на объект, т.е. воздействие окружающей среды, какова цель управления и какие средства воздействия на объект имеются в распоряжение системы управления.

Масштаб реального времени

Основным параметром, определяющим скорости всех процессов в системе управления, является время переходного процесса в объекте управления (tp). Оно определяет, какой масштаб времени следует выбрать при анализе и синтезе системы.

Например, для системы управления самолетом такой единицей временной шкалы является доля секунды, для системы управления предприятием – часы сутки.

Особенности работы СОД в масштабе реального времени:

чрезвычайно малое время, отведенное для принятия решения, соизмеримое со временем переходного процесса в объекте

сложность алгоритмов решения задач управления и практически мгновенное использование результатов решения для управления

недопустимость, как преждевременной выдачи управляющих сигналов, так и их запаздывания

Время в СОД

При работе системы в РВ используется либо астрономическое, либо относительное время.

Системы, работающие в астрономическом времени, несколько сложнее, систем, работающих с относительным временем. Это связано, с необходимостью периодически сверять астрономическое время с источником точного времени. Учет астрономического времени нужен при решении задач навигации, слежения за планетами и космическими объектами.

Относительное время широко используется в системах, которые управляют технологическим процессом и техническими системами. Это время отсчитывается либо с момента включения системы, либо с начала какой, либо фазы технологического процесса.

Системой реального времени называют аппаратно-программный комплекс, реагирующий в предсказуемые моменты времени на не предсказуемый поток внешних событий.

система должна успевать отреагировать на события, произошедшие на объекте, в течение времени, критического для этого события. Величина критического времени для каждого события определяется объектом и самим событием, и может быть разной, но время реакции системы должно быть предсказано при создании системы; отсутствие реакции в предсказанное время считается ошибкой для СРВ;

система должна успевать реагировать на одновременно происходящие события; даже если два или больше внешних событий происходит одновременно, система должна успеть среагировать на каждое из них в течении интервалов времени, критических для этих событий.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Что такое СОД?
Что такое СРВ?
Определение масштаба времени для СРВ?

Лекция 2

Требования к операционным системам реального времени

ОСРВ должна быть многонитевой (многопоточной) и прерываемой.

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.

ОСРВ должна обладать понятием приоритета для потоков.

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.

ОСРВ должна поддерживать предсказуемые механизмы синхронизации.

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

ОСРВ должна обеспечивать механизм наследования приоритетов.

Комбинация приоритета нити и разделение ресурсов между ними приводит к другому явлению: классической проблеме инверсии приоритетов. 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).

Характеристики ОСРВ

Время реакции системы на внешние события.

Согласно определению, ОСРВ должна обеспечить требуемый уровень сервиса в заданный промежуток времени. Этот промежуток времени задается обычно периодичностью и скоростью процессов, которым управляет система.

Приблизительное время реакции в зависимости от области применения ОСРВ может быть следующее:

• математическое моделирование- несколько микросекунд
• радиолокация- несколько миллисекунд
• складской учет- несколько секунд
• управление производством - несколько минут

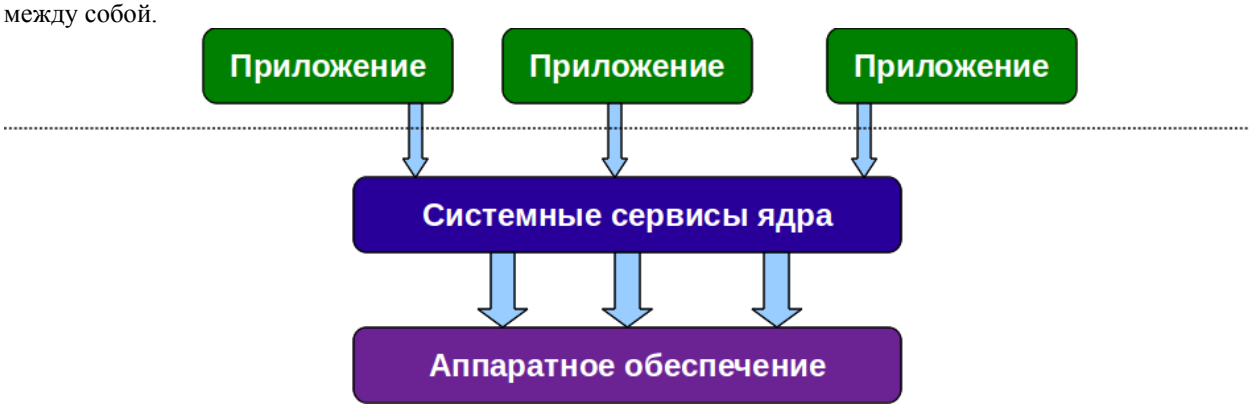
Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Назовите основные требования к ОСРВ.
Перечислите основные характеристики ОСРВ.

Лекция 3

Архитектура операционных системам реального времени

Монолитная архитектура. ОС определяется как набор модулей, взаимодействующих между собой внутри ядра системы и предоставляющих прикладному ПО входные интерфейсы для обращений к аппаратуре. Недостатки: плохая предсказуемость поведения, вызванная сложным взаимодействием модулей между собой.



Уровневая (слоевая) архитектура Прикладное ПО имеет возможность получить доступ к аппаратуре не только через ядро системы и её сервисы, но и напрямую. По сравнению с монолитной такая архитектура обеспечивает значительно большую степень предсказуемости реакций системы, а также позволяет осуществлять быстрый доступ прикладных приложений к аппаратуре. Главным недостатком таких систем является отсутствие многозадачности.

Архитектура «клиент-сервер» Основной принцип архитектуры заключается в вынесении сервисов ОС в виде серверов на уровень пользователя и выполнении микроядром функций диспетчера сообщений между клиентскими пользовательскими программами и серверами — системными сервисами.

Объектная архитектура на основе объектов-микроядер. В этой архитектуре API отсутствует вообще. Взаимодействие между компонентами системы (микроядрами) и пользовательскими процессами осуществляется посредством обычного вызова функций, поскольку и система, и приложения написаны на одном языке (для ОСРВ SoftKernel это C++).Это обеспечивает максимальную скорость системных вызовов.

Функции ядра ОСРВ

Синхронизация ресурсов. Метод синхронизации требует ограничить доступ к общим ресурсам (данным и внешним устройствам). Наиболее распространенный тип примитивной синхронизации -

двоичный семафор, обеспечивающий избирательный доступ к общим ресурсам. Так, процесс, требующий защищенного семафором ресурса, вынужден ожидать до тех
пор, пока семафор не станет доступным, что свидетельствует об освобождении ожидаемого ресурса, и, захватив ресурс, установить семафор. В свою очередь, другие процессы также будут ожидать доступа к ресурсу вплоть до того момента, когда семафор возвратит соответствующий ресурс системе распределения ресурсов.

Системы, обладающие большей ошибкоустойчивостью могут иметь счетный семафор. Этот вид семафора разрешает одновременный доступ к ресурсу лишь определенному количеству процессов.

Межзадачный обмен. Часто необходимо обеспечить передачу данных между программами внутри одной и той же системы Кроме того, во многих приложениях возникает необходимость взаимодействия с другими системами через сеть.

Внутренняя связь может быть осуществлена через систему передачи сообщений.

Внешнюю связь можно организовать либо через датаграмму (наилучший способ доставки), либо по линиям связи (гарантированная доставка). Выбор того или иного способа зависит от протокола связи.

Разделение данных. В прикладных программах, работающих в реальном времени, наиболее длительным является сбор данных. Данные часто необходимы для работы других программ или нужны системе для выполнения каких-либо своих функций.

Во многих системах предусмотрен доступ к общим разделам памяти. Широко распространена организация очереди данных. Применяется много типов очередей, каждый из которых обладает собственными достоинствами.

Обработка запросов внешних устройств. Каждая прикладная программа в реальном времени связана с внешним устройством определенного типа. Ядро должно обеспечивать службы ввода/вывода, позволяющие прикладным программам осуществлять чтение с этих устройств и запись на них.

Для приложений реального времени обычным является наличие специфического для данного приложения внешнего устройства. Ядро должно предоставлять сервис, облегчающий работу с драйверами устройств. Например, давать возможность записи на языках высокого уровня - таких, как Си или Паскаль.

Обработка особых ситуаций. Особая ситуация представляет собой событие, возникающее во время выполнения программы. Она может быть синхронной, если ее возникновение предсказуемо, как, например, деление на нуль. А может быть и асинхронной, если возникает непредсказуемо, как, например, падение напряжения.

Предоставление возможности обрабатывать события такого типа позволяет прикладным программам реального времени быстро и предсказуемо отвечать на внутренние и внешние события.

Существуют два метода обработки особых ситуаций - использование значений состояния для обнаружения ошибочных условий и использование обработчика особых ситуаций для прерывания ошибочных условий и их корректировки.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Каковы основные особенности монолитной архитектуры?

Каковы основные особенности слоеной архитектуры?

Каковы основные особенности архитектуры клиент-сервер?

Лекция 4

Стандарты ОСРВ

POSIX

(IEEE Portable Operating System Interface for Computer Environments, IEEE 1003.1) Стандарт POSIX был создан как стандартный интерфейс сервисов операционных систем.

Этот стандарт дает возможность создавать переносимые приложения. Впоследствии этот стандарт был расширен особенностями режима реального времени

Несмотря на то, что стандарт POSIX вырос из Unix, он затрагивает основополагающие абстракции операционных систем, а расширения реального времени применимы ко всем ОСРВ.

К настоящему времени стандарт POSIX рассматривается как семейство родственных стандартов: IEEE Std 1003.n (где n – это номер).

Стандарт 1003.1b (Realtime Extensions) содержит расширения реального времени:

В. Блокирование виртуальной памяти. Хотя в базовом стандарте POSIX использование виртуальной памяти не требуется, в UNIX-системах этот механизм широко распространен. Он очень эффективен при работе программ, не относящихся к программам реального времени, но приводит к непредсказуемости времени реакции системы.

Для того чтобы ограничить время доступа к памяти, в стандарте 1003.1b определяются функции блокировки (фиксации) в памяти всего адресного пространства процесса или отдельных его областей.

Эти функции следует использовать для критичных ко времени процессов, а также для процессов, с которыми синхронизируются критичные ко времени процессы. В этом случае время их реакции может быть предсказуемым.

C. Синхронизация процессов. В стандарте 1003.1b определяются функции управления синхронизацией процессов с помощью семафоров-счетчиков.

Эти семафоры идентифицируются по имени, находящемся в некотором пространстве имен, определяемом при реализации стандарта. Это пространство имен может совпадать, но не обязательно, с пространством имен файлов.

Семафор-счетчик – это общий механизм синхронизации, который позволяет реализовать взаимно исключающий доступ к разделяемым ресурсам, передачу сигналов, ожидание процессов и другие механизмы синхронизации.

D. Разделяемая память. В соответствии с базовым стандартом POSIX процессы имеют независимые адресные пространства, но во многих приложениях реального времени (и других) требуется совместное использование, с очень малыми издержками, большого количества данных.

Это возможно в случае, если процессы могут разделять части физической памяти. В стандарте 1003.1b определяются объекты разделяемой памяти

E. Сигналы реального времени. Сигналы реального времени устанавливаются в очередь, поэтому события не теряются. Необработанные сигналы реального времени извлекаются из очереди по приоритетам, где в качестве приоритета служит номер сигнала. Это предоставляет возможность быстрого отклика на события, требующие неотложной реакции.

Сигналы реального времени содержат дополнительное поле данных, которое может использоваться прикладной системой для обмена между генератором сигнала и его обработчиком. Например, это поле данных может использоваться для идентификации источника сигнала.

F. Взаимодействие процессов. Очереди сообщений идентифицируются по имени, принадлежащему некоторому пространству имен, определяемому при реализации стандарта.

Сообщения имеют связанное с ними поле приоритета и извлекаются из очереди в соответствии с приоритетом.

Передача и получение сообщений может блокироваться и разблокироваться.

Передача и получение не синхронизируются, то есть, отправитель не ждет, когда получатель действительно извлечет сообщение из очереди.

Максимальный размер сообщений и очередей определяется пользователем, а необходимые для поддержания очереди ресурсы могут выделяться заранее во время разработки приложения, что повышает предсказуемость работы с очередями сообщений.

G. Часы и таймеры. Часы реального времени должны обеспечивать разрешение минимум 20 мс.

Для отсчета временных интервалов на основе часов реального времени или других часов, определенных при реализации стандарта, могут создаваться таймеры.

По истечении заданного интервала времени эти таймеры генерируют сигнал, направленный процессу, создавшему данный таймер.

Существует несколько опций, таких, как периодическая сигнализация, единичный сигнал и т. д., которые позволяют легко реализовать, например, генерацию периодических событий.

H. Асинхронный ввод/вывод В стандарте 1003.1b определяются функции, которые обеспечивают возможность совмещать прикладную обработку и операции ввода/вывода, инициированные данным приложением.

Асинхронные операции ввода/вывода подобны обычным операциям, за исключением того, что после того как процесс инициировал асинхронную операцию ввода/вывода, он продолжает выполняться параллельно этой операции.

Когда операция завершается, данному приложению может быть послан сигнал.

DO-178B

Стандарт DO-178B, создан Радиотехнической комиссией по аэронавтике (RTCA, Radio Technical Commission for Aeronautics) для разработки ПО бортовых авиационных систем.

Стандартом предусмотрено пять уровней серьезности отказа, и для каждого из них определен набор требований к программному обеспечению, которые должны гарантировать работоспособность всей системы в целом при возникновении отказов данного уровня серьезности

Данный стандарт определяет следующие уровни сертификации:

- A (катастрофический),
- B (опасный),
- C (существенный),
- D (несущественный)
- E (не влияющий).

До тех пор пока все жесткие требования этого стандарта не будут выполнены, вычислительные системы, влияющие на безопасность, никогда не поднимутся в воздух.

ARINC-653

Стандарт ARINC-653 (Avionics Application Software Standard Interface) разработан компанией ARINC в 1997 г.

Этот стандарт определяет универсальный программный интерфейс APEX (Application/Executive) между ОС авиационного компьютера и прикладным ПО.

Требования к интерфейсу между прикладным ПО и сервисами операционной системы определяются таким образом, чтобы разрешить прикладному ПО контролировать диспетчеризацию, связь и состояние внутренних обрабатываемых элементов.

В 2003 г. принята новая редакция этого стандарта. ARINC-653 в качестве одного из основных требований для ОСРВ в авиации вводит архитектуру изолированных (partitioning) виртуальных машин.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Основные особенности стандарта POSIX для ОСРВ.

Лекция 5

Планирование задач

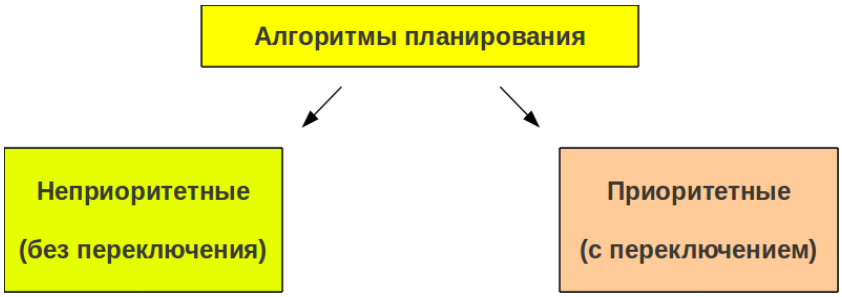
Необходимость планирования задач появляется, как только в очереди активных (готовых) задач появляются более одной задачи (в многопроцессорных системах - более числа имеющихся процессоров).

Алгоритм планирования задач является основным отличием систем реального времени от "обычных" операционных систем.

В ОС общего назначения целью планирования является обеспечение выполнения всех задач из очереди готовых задач, обеспечивая иллюзию их параллельной работы и не допуская монополизацию процессора какой-либо из задач.

В ОСРВ же целью планирования является обеспечение выполнения каждой готовой задачи к определенному моменту времени, при этом часто "параллельность" работы задач не допускается, поскольку тогда время исполнения задачи будет зависеть от наличия других задач.

Планировщик задач (scheduler) - это модуль (программа), отвечающий за разделение времени имеющихся процессоров между выполняющимися задачами. Отвечает за коммутацию задач из состояния блокировки в состояние готовности, и за выбор задачи (задач - по числу процессоров) из числа готовых для исполнения процессором (ами).



Ключевым вопросом планирования является выбор момента принятия решения

Приоритетом называется число, приписанное операционной системой (а именно, планировщиком задач) каждому процессу и задаче.

Существуют несколько схем назначения приоритетов.

Фиксированные приоритеты - приоритет задаче назначается при ее создании и не меняется в течение ее жизни.

Эта схема с различными дополнениями применяется в большинстве систем реального времени. В схемах планирования ОСРВ часто требуется, чтобы приоритет каждой задачи был уникальным.

Турнирное определение приоритета - приоритет последней исполнявшейся задачи понижается.

Определение приоритета по алгоритму round robin - приоритет задачи определяется ее начальным приоритетом и временем ее обслуживания.

Чем больше задача обслуживается процессором, тем меньше ее приоритет (но не опускается ниже некоторого порогового значения).

Эта схема в том или ином виде применяется в большинстве UNIX систем.

Алгоритм диспетчеризации FIFO

Является алгоритмом планирования без переключений. Процессам предоставляется доступ к процессору в том порядке, в котором они его запрашивают.

При FIFO диспетчеризации процесс продолжает выполнение, пока не наступит момент, когда он:

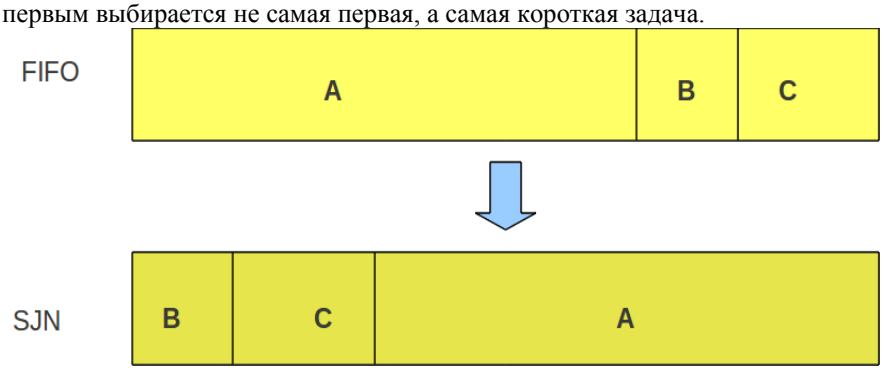
• добровольно уступает управление (заканчивается, блокируется и т.п.);
• вытесняется процессом с более высоким приоритетом.

При отсутствии второго условия возможен случай, когда высокоприоритетная задача будет ожидать окончания работы низкоприоритетной.

«Кратчайшая задача – первая»

Shortest job next (SJN) Shortest Job First (SJF) Shortest Process Next (SPN)

Этот алгоритм без переключений предполагает, что временные отрезки работы известны заранее. В этом алгоритме первым выбирается не самая первая, а самая короткая задача.



Наименьшее оставшееся время выполнения

В соответствии с этим алгоритмом планировщик каждый раз выбирает процесс с наименьшим оставшимся временем выполнения.

В этом случае также необходимо знать заранее время выполнения каждого процесса.

Когда поступает новый процесс, его полное время выполнения сравнивается с оставшимся временем выполнения текущего процесса. Если время выполнения нового процесса меньше, текущий процесс приостанавливается и управление передается новому процессу.

Эта схема позволяет быстро обслуживать короткие процессы.

«Карусельная диспетчеризация (циклическое планирование)».

При карусельной диспетчеризации процесс продолжает выполнение, пока не наступит момент, когда он:
• добровольно уступает управление (т.е. блокируется);
• вытесняется процессом с более высоким приоритетом;
• использовал свой квант времени (timeslice).

После того, как процесс использовал свой квант времени, управление передается следующему процессу, который находится в состоянии готовности и имеет такой же уровень приоритета.

«Адаптивная диспетчеризация»

При адаптивной диспетчеризации процесс ведет себя следующим образом:

Если процесс использовал свой квант времени (т.е. он не блокировался), то его приоритет уменьшается на 1. Это получило название снижение приоритета (priority decay).

"Пониженный" процесс не будет продолжать "снижаться", даже если он использовал еще один квант времени и не блокировался - он снизится только на один уровень ниже своего исходного приоритета.

Если процесс блокируется, то ему возвращается первоначальное значение приоритета.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Перечислите основные методы диспетчеризации и их особенности.

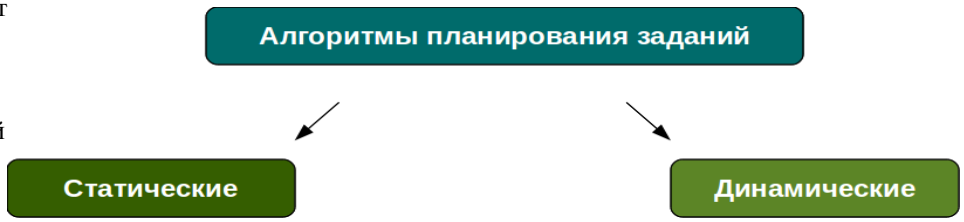
Лекция 6

Планирование периодических процессов

Внешние события, на которые система реального времени должна реагировать, можно разделить на периодические (возникающие через регулярные промежутки времени) и непериодические (возникающие непредсказуемо).

Возможно наличие нескольких потоков событий, которые система должна обрабатывать. В зависимости от времени, затрачиваемого на обработку каждого из событий, может оказаться, что система не в состоянии своевременно обработать все события.

Если в систему поступает m периодических событий, событие с номером i поступает с периодом Pi и на его обработку уходит Ci секунд работы процессора, все потоки могут быть своевременно обработаны только при выполнении условия



Статические алгоритмы определяют приемлемый план выполнения заданий по их априорным характеристикам, динамический алгоритм модифицирует план во время исполнения заданий.

Издержки на статическое планирование низки, но оно крайне нечувствительно и требует полной предсказуемости той системы реального времени, на которой оно установлено.

Динамическое планирование связано с большими издержками, но способно адаптироваться к меняющемуся окружению.

Алгоритм RMS (Rate Monotonic Scheduling)

Статический алгоритм планирования реального времени для прерываемых периодических процессов - алгоритм RMS (Rate Monotonic Scheduling – планирование с приоритетом, пропорциональным частоте).

Этот алгоритм может использоваться для процессов, удовлетворяющих следующим условиям:

1. Каждый периодический процесс должен быть завершен за время его периода

2. Ни один процесс не должен зависеть от любого другого процесса

3. Каждому процессу требуется одинаковое процессорное время на каждом интервале

4. У непериодических процессов нет жестких сроков

5. Прерывание процесса происходит мгновенно, без накладных расходов.

Алгоритм EDF

(Earliest Deadline First – процесс с ближайшим сроком завершения в первую очередь).

Алгоритм EDF представляет собой динамический алгоритм, не требующий от процессов периодичности. Он также не требует и постоянства временных интервалов использования процессора.

Каждый раз, когда процессу требуется процессорное время, он объявляет о своем присутствии и о своем сроке выполнения задания. Планировщик хранит список процессов, сортированный по срокам выполнения заданий.

Алгоритм запускает первый процесс в списке, то есть тот, у которого самый близкий по времени срок выполнения.

Когда новый процесс переходит в состояние готовности, система сравнивает его срок выполнения со сроком выполнения текущего процесса.

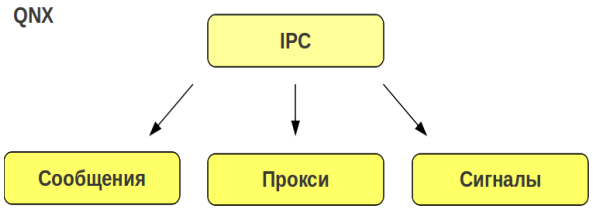
Если у нового процесса график более жесткий, он прерывает работу текущего процесса.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Перечислите основные методы диспетчеризации периодических процессов и их особенности.

Лекция 7

Межпроцессное взаимодействие - IPC



Сообщения

Сообщения в QNX - это пакеты байт, которые синхронно передаются от одного процесса к другому. QNX при этом не анализирует содержание сообщения. Передаваемые данные понятны только отправителю и получателю и никому более.

Для непосредственной связи друг с другом взаимодействующие процессы используют следующие функции:

Send() // посылка сообщений

Receive() // получение сообщений

Reply() //ответ процессу, пославшему сообщение

Эти функции могут быть использованы как локально, т.е. для связи между процессами на одном компьютере, так и в пределах сети, т.е. для связи между процессами на разных узлах.

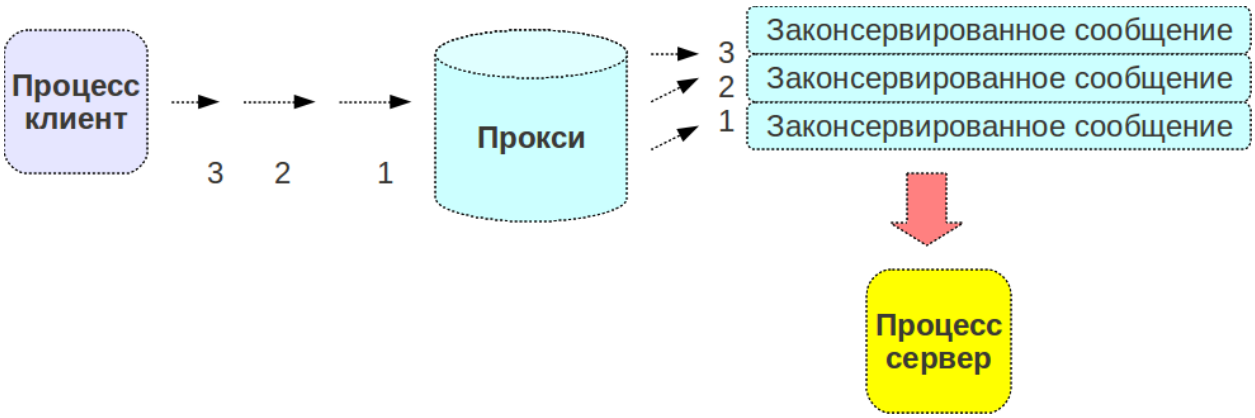
Прокси

Прокси - это форма неблокирующего сообщения, особенно подходящего для извещения о наступлении события, когда посылающий процесс не нуждается в диалоге с получателем.

Единственная функция прокси состоит в посылке фиксированного сообщения определенному процессу, который является владельцем прокси.

Подобно сообщениям, прокси могут быть использованы в пределах всей сети.

Используя прокси, процесс или обработчик прерывания может послать сообщение другому процессу, не блокируясь и не ожидая ответа.



Сигналы

Сигналы являются традиционным способом связи, которая используется в течение многих лет в различных операционных системах.

Сигналы во многом схожи с прерываниями. Они позволяют прервать выполнение прикладной программы и отреагировать на породившее сигнал событие.

Сигналы, в частности, используются

при обработке исключений (деление на 0, использование неверного адреса и др.);

для сообщения об асинхронном событии (об окончании операции ввода/вывода, срабатывании таймера и др.);

для организации взаимодействия потоков управления.

Сигналы могут генерироваться (посылаться процессу) как операционной системой, так и процессом.

В системе имеется несколько видов сигналов. Каждому сигналу соответствует уникальное положительное число (номер сигнала). Кроме того, для сигналов определены имена.

Реакция процесса на сигнал:. Если процесс не определил никаких специальных мер по обработке сигнала, то выполняется предусмотренное для сигнала действие по умолчанию – обычно таким действием по умолчанию является завершение работы процесса;

процесс может игнорировать сигнал. Если процесс игнорирует сигнал, то сигнал не оказывает на процесс никакого воздействия;

процесс может предусмотреть обработчик сигнала - функцию, которая будет вызываться при приеме сигнала. Если процесс содержит обработчик для какого-либо сигнала, говорят, что процесс может "поймать" этот сигнал. Любой процесс, который "ловит" сигнал, фактически получает особый вид программного прерывания. Никакие данные с сигналом не передаются.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

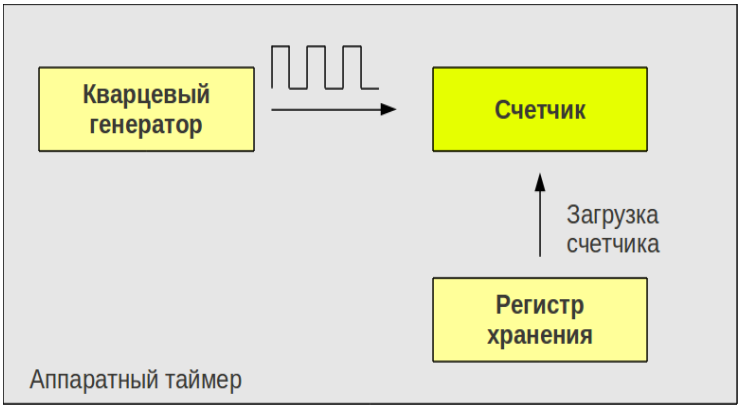
Перечислите основные методы IPC и их особенности.

Лекция 8

Время в ОСРВ

Фиксация временных интервалов (тайм-ауты, тайм-ауты ядра, интервальные таймеры и др.) и хронометраж выполнения участков кода для операционных систем реального времени на порядок более критичны, чем для операционных систем общего назначения.

Кроме того, время весьма важно при запуске задач по расписанию, а также при планировании задач.



У программируемого таймера обычно есть несколько режимов работы.

В режиме одновибратора при запуске таймера содержимое регистра хранения копируется в счетчик. Затем содержимое счетчика уменьшается на единицу при каждом импульсе от кристалла. Когда счетчик достигает нуля. он вызывает прерывание и останавливается до тех пор, пока не будет снова явно запущен программным обеспечением.

В режиме генератора прямоугольных импульсов при достижении счетчиком нуля также инициируется прерывание, но содержимое регистра хранения автоматически копируется в счетчик, и весь процесс повторяется снова бесконечно.

Преимущество программируемого таймера состоит в том, что частота прерываний от него может управляться программно.

Например, если используется кристалл с частотой колебаний 500 МГц, то счетчик получает импульс каждые 2 нс. При использовании 32-разрядного регистра можно запрограммировать возникновение прерываний через равные интервалы времени от 2 нс до 8.6 с, называемые тиками.

Микросхемы программируемых таймеров могут содержать несколько независимых программируемых таймеров.

Все, что делает таймер – это инициирует прерывания через определенные интервалы времени.

Стандарт POSIX определяет, что система всегда содержит, по крайней мере, одни часы с идентификатором CLOCK_REALTIME (системные часы).

Значение этих часов интерпретируется как календарное время, то есть время (в секундах и наносекундах), истекшее с 0 часов 1 января 1970 года.

При наличии соответствующего оборудования могут быть созданы дополнительные часы.

Функция clock_settime() позволяет установить показания часов, функция clock_gettime()- опросить показания часов, а clock_getres() - узнать разрешающую способность часов.

Все три функции работают с высокой точностью, так используют структуру timespec, которая позволяет хранить время в секундах и наносекундах.

Стандарт POSIX также определяет, каким образом можно программно создавать и использовать таймеры.

Для создания таймера используется функция timer_create(). Одним из аргументов этой функции является структура sigevent, которая определяет вид оповещения о срабатывании таймера (например, посылка сигнала или выполнение указанной функции).

Установка и запуск таймера производится функцией timer_settime(). Эта функция определяет время первого срабатывания таймера, а также период срабатывания (если требуется периодическое срабатывание таймера).

Модель временной шкалы

Модель временной шкалы в операционных системах реального времени основана на следующих предпосылках:

микроядро операционной системы «живет» в дискретной сетке времени;

каждый единичный момент времени для микроядра – это тик системного

времени;

какие-либо изменения состояний времени фиксируются микроядром только в узлах этой дискретной шкалы времени;

микроядро «не различает» два временных события, если они происходят

между соседними тиками системного времени (т.е. с интервалом меньшим, чем интервал системного тика).

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Особенности организации счета времени в ОСРВ.

Лекция 9

Обзор ОСРВ

RTLinux

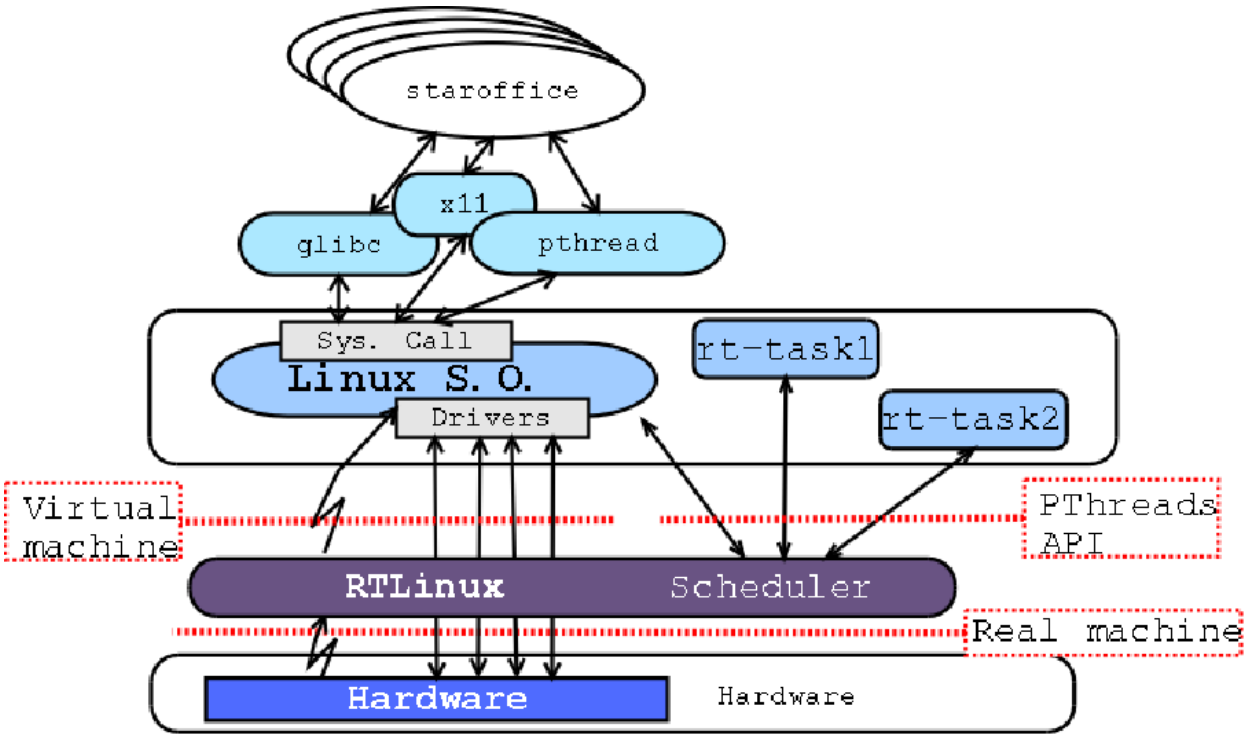
— микроядерная операционная система жёсткого реального времени, которая выполняет Linux как полностью вытесняемый процесс.

Разработчики RTLinux пошли по тому пути, который предусматривает запуск из наноядра реального времени ядра Linux как задачи с наименьшим приоритетом.

В RTLinux все прерывания обрабатываются ядром реального времени, которое включает собственный планировщик задач, обработчик прерываний и библиотечный код.

В случае отсутствия обработчика реального времени для какого-то прерывания, оно передаётся в Linux.

Фактически Linux является простаивающей (idle) задачей ОСРВ, запускаемой только в том случае, если никакая задача не исполняется в реальном времени.



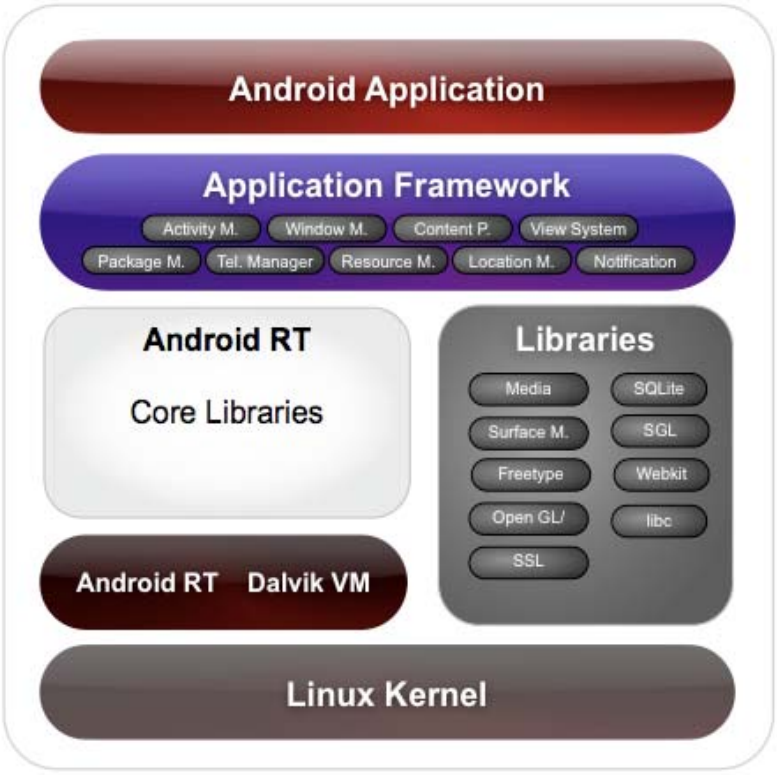
Android

— операционная система для коммуникаторов, планшетных компьютеров, цифровых проигрывателей, цифровых фоторамок, наручных часов, нетбуков и смартбуков, основанная на ядре Linux.

Изначально разрабатывалась компанией Android Inc., которую затем купила Google.

Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас и занимается поддержкой и дальнейшим развитием платформы.

Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit создаёт приложения, написанные на Си и других языках.

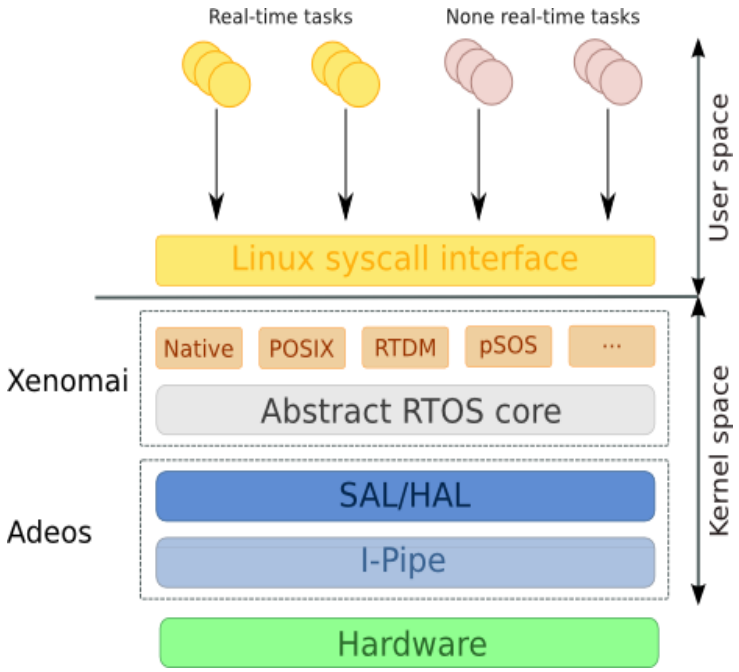


Xenomai

— это фреймворк для разработки приложений реального времени, связанный с ядром Linux, для того, чтобы предоставить всеобъемлющую, с открытым интерфейсом жёсткого реального времени поддержку приложений, легко интегрируемых в окружение Linux.

Проект Xenomai начался в августе 2001 года. В 2003-м он был объединён с проектом RTAI, чтобы предоставить свободную платформу промышленного уровня для Linux, названную RTAI/fusion, на базе ядра Xenomai для абстрактной операционной системы реального времени.

В конечном счёте проект RTAI/fusion стал независимым от RTAI в 2005 году под названием Xenomai.



QNX

- POSIX-совместимая операционная система реального времени, предназначенная преимущественно для встраиваемых систем. Считается одной из лучших реализаций концепции микроядерных операционных систем.

QNX основана на идее работы основной части своих компонентов, как небольших задач, называемых сервисами.

Использование микроядра в QNX позволяет пользователям (разработчикам) отключить любую ненужную им функциональность, не изменяя ядро. Вместо этого можно просто не запускать определённый процесс.

Windows CE

- это вариант операционной системы Microsoft Windows для наладонных компьютеров, мобильных телефонов и встраиваемых систем.

Сегодня Windows CE (Consumer Electronics — бытовая техника) не является «урезанной» версией Windows для настольных ПК, она основана на совершенно другом ядре и является операционной системой реального времени с набором приложений, основанных на Microsoft Win32 API.

Windows CE — это компонентная, многозадачная, многопоточная, многоплатформенная операционная система с поддержкой реального времени. Разработчикам доступны около 600 компонентов, используя которые они могут создавать собственные образы операционной системы, которые включает только необходимый данному конкретному устройству функционал.

Windows CE оптимизирована для устройств, имеющих минимальный объём памяти: ядро Windows CE может работать на 32 КБ памяти. С графическим интерфейсом (GWES) для работы Windows CE понадобится от 5 МБ. Устройства часто не имеют дисковой памяти и могут быть сконструированы как «закрытые» устройства, без возможности расширения пользователем (например, ОС может быть «зашита» в ПЗУ).

VxWorks

- операционная система реального времени, разрабатываемая компанией Wind River Systems (США) (приобретена компанией Intel 17 июля 2009 г.), ориентированная на использование в встраиваемых компьютерах, работающих в системах жёсткого реального времени.

VxWorks является системой с кросс-средствами разработки прикладного программного обеспечения. Иначе говоря, разработка происходит на инструментальном компьютере, называемом host, для последующего применения его на целевой машине — target.

VxWorks имеет архитектуру клиент-сервер и, как и большинство ОС жёсткого реального времени, построена по технологии микроядра.

На самом нижнем непрерываемом уровне ядра (WIND Microkernel) выполняются только базовые функции планирования задач и управления коммуникацией/синхронизацией между задачами. Все остальные функции ОСРВ более высокого уровня — управление памятью, сетевые средства и т. д. — реализуются через простые функции нижнего уровня. За счёт такой иерархической организации достигается быстродействие и детерминированность ядра системы, также это позволяет легко строить необходимую конфигурацию операционной.

Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Особенности архитектуры RTLinux/.

Особенности архитектуры Xenomai.

Особенности архитектуры QNX/.

Особенности архитектуры Windows CE/.

Лекция 10

SCADA

Supervisory Control And Data Acquisition, Диспетчерское управление и сбор данных — программный пакет предназначенный для разработки или обеспечения работы в реальном времени систем сбора, обработки, отображения и архивирования информации об объекте мониторинга или управления.

SCADA-системы используются там, где требуется обеспечивать операторский контроль за технологическими процессами в реальном времени.

Термин SCADA имеет двоякое толкование. Наиболее широко распространено понимание SCADA как приложения, то есть программного комплекса, обеспечивающего выполнение указанных функций, а также инструментальных средств для разработки этого программного обеспечения.

Однако, часто под SCADA-системой подразумевают программно-аппаратный комплекс.

В 80-е годы под SCADA-системами чаще понимали программно-аппаратные комплексы сбора данных реального времени. С 90-х годов термин SCADA больше используется для обозначения только программной части человеко-машинного интерфейса (HMI).

SCADA - задачи

Обмен данными с промышленными контроллерами и платами ввода/вывода в реальном времени через драйверы.

Обработка информации в реальном времени.

Логическое управление.

Отображение информации на экране монитора в удобной и понятной для человека форме.

Ведение базы данных реального времени с технологической информацией.

Аварийная сигнализация и управление тревожными сообщениями.

Подготовка и генерирование отчетов о ходе технологического процесса.

Осуществление сетевого взаимодействия между SCADA ПК.

Обеспечение связи с внешними приложениями.

SCADA - PLC

Программируемый логический контроллер (ПЛК) или Programmable Logic Controller (PLC) или программируемый контроллер — электронная составляющая промышленного контроллера, используемого для автоматизации технологических процессов.

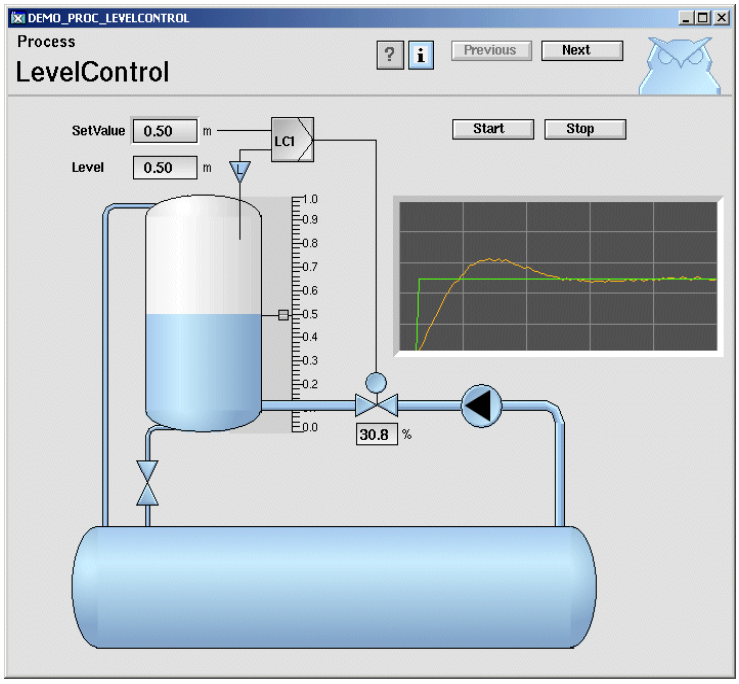
Условия работы ПЛК:

неблагоприятная окружающая среда,

автономное использование, без серьёзного обслуживания и практически без вмешательства человека.

ПЛК являются устройствами реального времени

SCADA — интерфейс пользователя



Теоретические вопросы, упражнения, задачи и задания для самоконтроля

Что такое SCADA?

Задачи SCADA?

Что такое PLC?

.

Список литературы

Ослэндер Д. М., Риджли Дж. Р., Рингенберг Дж. Д. Управляющие программы для механических систем: Объектно-ориентированное проектирование систем реального времени — М.: Бином. Лаборатория знаний, 2004. — 416 с.

Кёртен Р. Введение в QNX/Neutrino 2 — СПб.: Петрополис, 2001. — 512 с.

Зыль С. QNX Momentics. Основы применения — СПб.: БХВ-Петербург, 2004. — 256 с.

Сидельников В. В., Широков В. В. Управление процессами в программных средах АСОИУ: Учеб. пособие / ГЭТУ.- С.-Пб., 1994. –64 с.

Зыль С. Операционная система реального времени QNX: от теории к практике — 2-е изд. — СПб.: БХВ-Петербург, 2004. — 192 с.

Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. – М.: ДМК,