ГЛАВА 13

Встроенные операционные системы

В ЭТОЙ ГЛАВЕ...

Учебные цели

- Разъяснить понятие встроенной системы.
- Уяснить характеристики встроенных операционных систем.
- Объяснить отличия обычной ОС Linux от встроенной.
- Описать архитектуру и основные функциональные возможности операционной системы TinyOS.

В этой главе рассматриваются встроенные операционные системы — одна из самых важных и широко употребляемых категорий операционных систем. Среда встроенной системы накладывает на операционную систему особые жесткие требования; кроме того, предусматриваются совсем иные стратегии проектирования, чем в случае обычных операционных систем.

Начнем эту главу с краткого обзора понятия встроенных систем, а затем перейдем к рассмотрению принципов действия встроенных операционных систем. И наконец в этой главе будут представлены два самых разных подхода к проектированию встроенных операционных систем: встроенной операционной системы Linux и операционной системы TinyOS. Еще один подход к проектированию встроенных операционных систем рассматривается в приложении P, “eCos”, на примере eCos — еще одной очень важной встроенной ОС.

13.1. Встроенные системы

В этом разделе представлено понятие встроенной системы. Для понимания этой концепции требуется также пояснить, чем микропроцессор отличается от микроконтроллера.

Концепции встроенных систем

Термин *встроенная система* означает применение в конкретном продукте аппаратных и программных средств, выполняющих особую функцию или ряд функций, в отличие от универсального (например, переносного или настольного) компьютера. Встроенную систему можно также определить как любое устройство, в состав которого входит компьютерная микросхема, но не являющееся универсальной рабочей станцией, настольным или переносным компьютером. С одной стороны, ежегодно продаются сотни миллионов компьютеров, в том числе переносных и персональных, рабочих станций, серверов, больших и суперЭВМ. С другой стороны, ежегодно производятся десятки миллиардов микроконтроллеров, встраиваемых в более крупные устройства. Ныне большинство устройств, работающих на электрической энергии, содержат встроенную вычислительную систему. И, вероятнее всего, в ближайшем будущем практически все подобные устройства будут иметь в своем составе встроенные вычислительные системы.

Типы устройств со встроенными системами слишком многочисленны, чтобы пытаться их перечислить. К их числу относятся сотовые телефоны, цифровые фото- и видеокамеры, калькуляторы, микроволновые печи, домашние системы безопасности, стиральные и посудомоечные машины, системы освещения, термостаты, печатающие ус-

тройства, различные автомобильные системы (например, управления коробкой передач, регулирования скорости движения, впрыскивания топлива, незаклинивающих тормозов и подвесок), теннисные ракетки, зубные щетки и многочисленные типы датчиков и исполнительных механизмов в автоматизированных системах.

Нередко встроенные системы тесно связаны со своим окружением. А это может привести к ограничениям реального времени, накладываемым в связи с необходимостью взаимодействовать с окружающей средой. И такие ограничения (например, накладываемые скоростью движения, точностью измерения или временны*ы*ми промежутками) требуют синхронизации программных операций. Если же под контролем одновременно должны выполняться многие действия, то это накладывает еще более сложные ограничения — реального времени.

На рис. 13.1 наглядно показана организация встроенных систем с использованием наиболее общей терминологии.

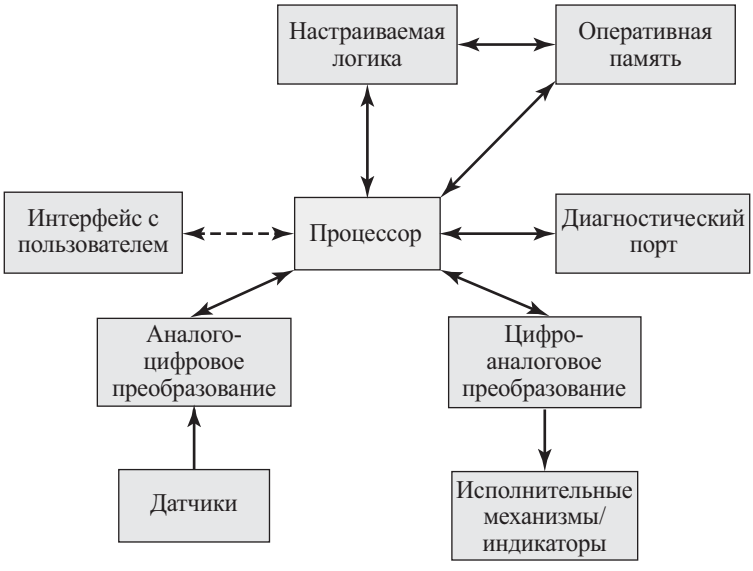

Рис. 13.1. Возможная организация встроенной системы

Типичный настольный компьютер отличается от переносного не только процессором и оперативной памятью, но и, как поясняется ниже, рядом других элементов.

- Для целей измерения, манипулирования или иного взаимодействия с внешним окружением в системе могут присутствовать самые разные интерфейсы. Встроенные системы нередко взаимодействуют (реагируют, манипулируют и сообщаются) с внешним миром через датчики и исполнительные механизмы, а следовательно, они, как правило, являются реагирующими системами. Реагирующая система находится в постоянном взаимодействии со своим окружением, выполняя действия в темпе, который задается данным окружением.
- Интерфейс пользователя может быть как простым (наподобие сигнальных лампочек), так и сложным (типа робототехнического зрения реального времени). Зачастую интерфейс с пользователем может просто отсутствовать.

- Диагностический порт может быть использован для диагностирования контролируемой системы (а не просто для работы компьютера).
- Для повышения производительности или надежности может использоваться интегральная схема специального назначения — программируемая (FPGA) или созданная для конкретного приложения (ASIC) или даже нецифровая аппаратура.
- Программное обеспечение нередко выполняет фиксированную функцию и предназначено для конкретного приложения.
- Первостепенное значение для встроенных систем имеет эффективность. Такие системы оптимизированы по потребляемой энергии, объему кода, времени выполнения, весу, габаритам и стоимости.

Имеется несколько заслуживающих внимания областей применения встроенных систем, сходных с универсальными вычислительными системами, как описывается ниже.

- Несмотря на фиксированную функциональность программного обеспечения, возможность обновления для устранения программных ошибок, повышения безопасности и расширения функциональных возможностей стала очень важным свойством встроенных систем, и не только в бытовой технике.
- Одной относительно недавней разработкой встроенных систем стали платформы, поддерживающие обширный ряд приложений. Характерными тому примерами служат смартфоны и аудиовизуальные устройства вроде “интеллектуальных” телевизоров.

Прикладные и специализированные процессоры

Прикладные процессоры (application processors) определяются как способные выполнять сложные операционные системы, подобные Linux, Android и Chrome. Таким образом, прикладной процессор имеет универсальный характер. Соответствующая встроенная система разработана для поддержки многочисленных приложений и выполнения обширного ряда функций.

В большинстве встроенных систем применяется **специализированный процессор** (dedicated processor), который, как подразумевает его название, специально предназначен для решения одной конкретной задачи или небольшого числа подобных задач по требованию главного устройства. А поскольку такая встроенная система изначально предназначена для решения конкретной задачи или нескольких задач, то процессор и связанные с ним компоненты могут быть спроектированы специальным образом, позволяющим сократить его габариты и стоимость.

Микропроцессоры

Микропроцессор — это процессор, элементы которого сведены в одну или несколько интегральных схем (ИС). Первоначально процессорные ИС включали в себя регистры, арифметико-логическое устройство (АЛУ) и некоторый блок управления или логику обработки команд. По мере увеличения плотности размещения транзисторов на кристалле появилась возможность усложнить архитектуру набора команд, а в конечном счете — добавить больше оперативной памяти и процессоров. Современные микропроцессорные ИС содержат несколько процессоров, иначе называемых *ядрами*, а также значительный

объем сверхоперативной или так называемой *кеш-памяти*. Но, как показано на рис. 13.2, микропроцессорная ИС включает в себя лишь некоторые элементы, образующие вычислительную систему.

Большинство компьютеров, включая встроенные компьютеры в смартфонах и планшетах, а также персональные и переносные компьютеры и рабочие станции, располагаются на материнской плате. Но, прежде чем рассматривать это размещение, необходимо определить ряд терминов. В частности, **печатная плата** (printed circuit board — PCB) представляет собой жесткую монтажную плоскость для установки и соединения ИС и прочих электронных компонентов. Такая плата состоит из нескольких слоев (как правило, от двух до десяти), соединяющих электронные компоненты с помощью медных дорожек, вытравленных на плате.

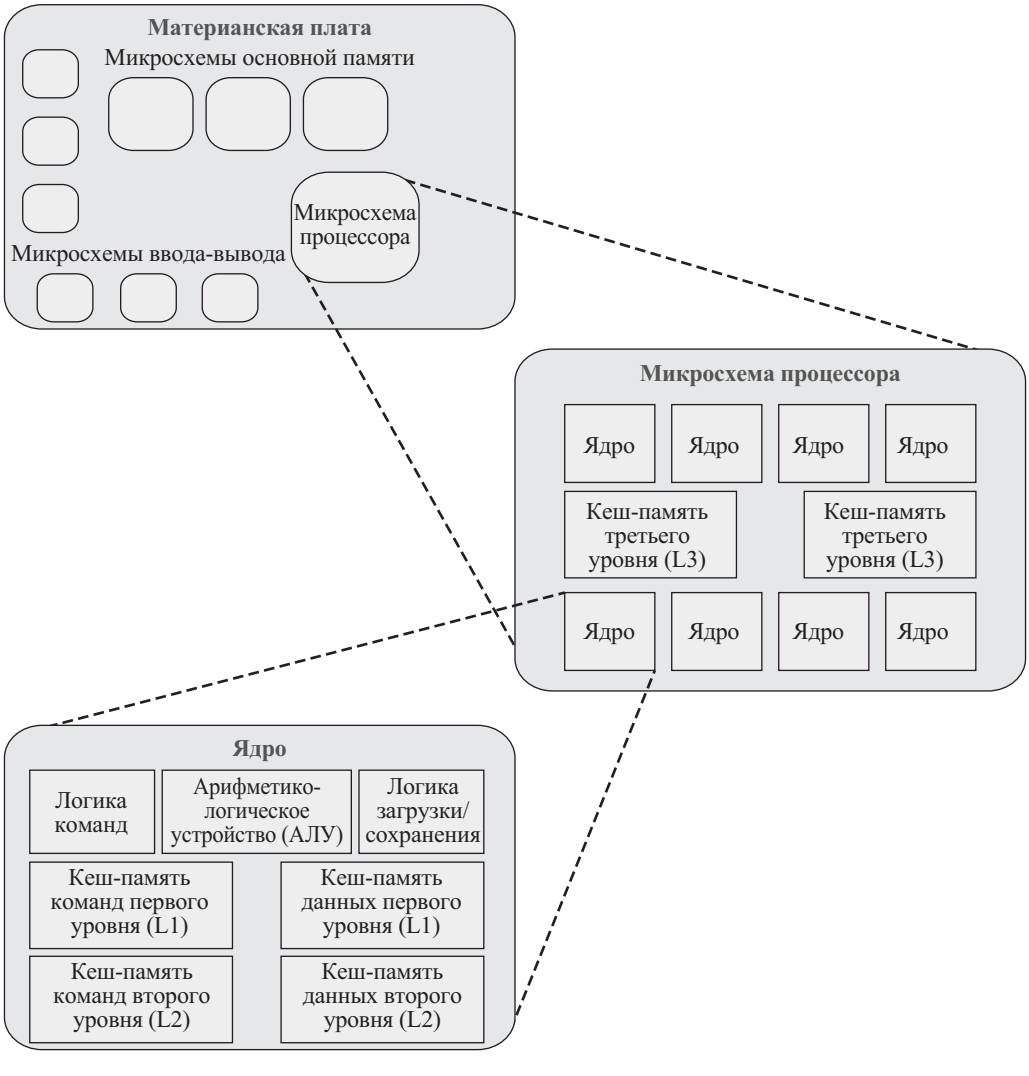

Рис. 13.2. Упрощенное схематическое представление основных элементов многоядерного компьютера

Основная печатная плата в компьютере называется **системной** или **материнской**, тогда как более мелкие печатные платы, вставляемые в разъемы на основной плате, называются платами расширения.

Самыми заметными элементами на системной плате являются микросхемы. **Микросхема** (чип) — это единое полупроводниковое (как правило, кремниевое) изделие с электронными и логическими схемами. Получающееся в конечном итоге изделие называется *интегральной схемой* (ИС).

На системной плате находится разъем или панель для установки процессора, который обычно состоит их нескольких отдельных ядер, и поэтому он называется *многоядерным процессором*. Там же находятся разъемы для установки интегральных схем оперативной памяти, контроллеров ввода-вывода и других базовых компонентов компьютера. В соответствующие разъемы системных плат настольных компьютеров можно установить дополнительные компоненты и платы расширения. Таким образом, современная материнская плата соединяет лишь некоторые отдельные микросхемы, причем каждая из них содержит от нескольких тысяч до сотен миллионов транзисторов.

Микроконтроллеры

Микроконтроллер — это однокристальная интегральная схема, состоящая из процессора, постоянной памяти для хранения программы (ПЗУ или флеш-памяти), кратковременной памяти (ОЗУ) для ввода-вывода данных, генератора тактовых сигналов и блока управления вводом-выводом. Иногда она называется “однокристальным компьютером”. Схемы микроконтроллеров используют доступное логическое пространство совершенно иначе. На рис. 13.3 представлена общая терминология для обозначения тех элементов, которые обычно находятся в интегральной схеме микроконтроллера. В частности, процессорная часть микроконтроллера занимает намного меньше места на кристалле, чем у других микропроцессоров, и обладает намного более высокой энергетической эффективностью.

Ежегодно миллиарды микроконтроллерных устройств встраиваются в миллиарды изделий: от игрушек до бытовых приборов и автомобилей. Например, в одном автомобиле может быть использовано 70 и более микроконтроллеров. Как правило, микроконтроллеры (особенно более мелкие и менее дорогие их разновидности) применяются в качестве специализированных процессоров, предназначенных для решения конкретных задач. Например, микроконтроллеры широко применяются в автоматизации производственных процессов. Обеспечивая простую реакцию на ввод, они способны управлять механизмами, включать и выключать вентиляторы, открывать и закрывать клапаны и т.д. Они являются неотъемлемой частью современной промышленной технологии и относятся к числу самых недорогих средств производства механизмов, способных выполнять чрезвычайно сложные функции.

Микроконтроллеры отличаются самыми разными габаритами и производительностью, а процессоры — архитектурами: от 4- до 32-разрядных. Микроконтроллеры обычно работают намного медленнее, чем микропроцессоры: первые, как правило, — в диапазоне тактовых частот порядка мегагерц, тогда как вторые — в диапазоне тактовых частот порядка гигагерц. Для микроконтроллеров также характерно, что они не обеспечивают взаимодействие с человеком. Микроконтроллер, встраиваемый в устройство, программируется для решения конкретной задачи, выполняемой данным устройством, а следовательно, вступает в действие, когда в этом возникает потребность.

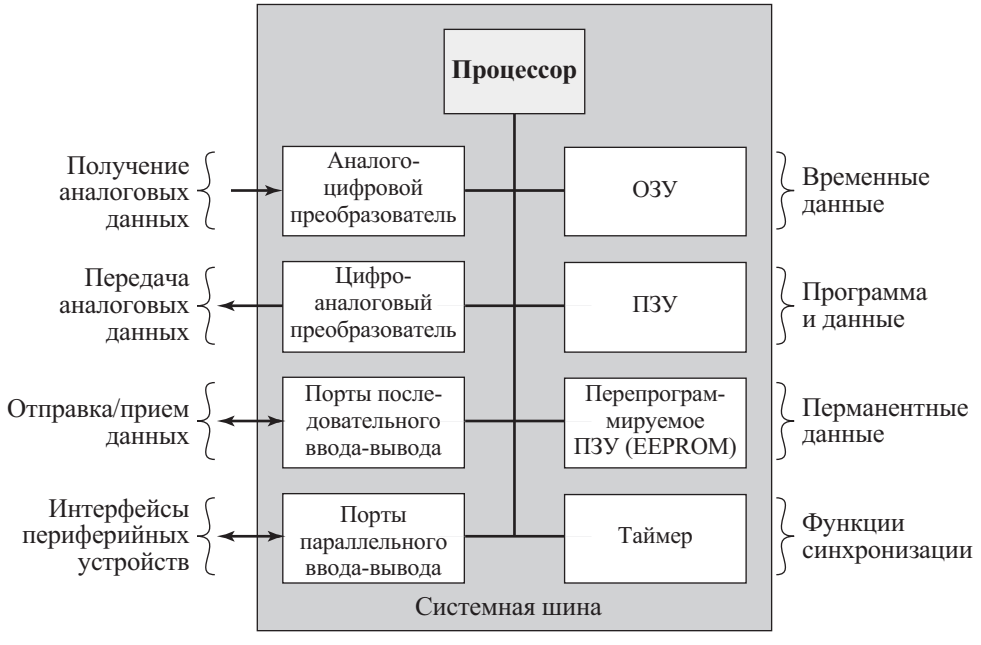

Рис. 13.3. Типичные элементы в ИС микроконтроллера

Глубоко встроенные системы

Б*о*льшая часть из общего числа встроенных систем называется **глубоко встроенными системами**. И хотя этот термин широко употребляется в технической и коммерческой литературе, искать ясное его определение в Интернете бесполезно (по крайней мере, автору этих строк найти такое определение не удалось). Как правило, можно сказать, что глубоко встроенная система содержит процессор, поведение которого трудно наблюдать как программисту, так и пользователю. В глубоко встроенной системе применяется микроконтроллер, а не микропроцессор, она программируется лишь один раз, когда программная логика работы конкретного устройства "зашивается" в ПЗУ, т.е. в доступную только для чтения память, а кроме того, она никак не взаимодействует с пользователем.

Глубоко встроенные системы являются специализированными устройствами узкого назначения, обнаруживающими нечто в своем окружении, выполняющими обработку сначала на элементарном уровне, а затем и получаемых результатов тем или иным образом. Глубоко встроенные системы нередко обладают возможностями беспроводной связи и присутствуют в таких сетевых конфигурациях, как сети датчиков, разворачиваемые на большой площади (например, на фабрике или сельскохозяйственном поле). В частности, Интернет вещей сильно зависит от глубоко встроенных систем. Как правило, в глубоко встроенных системах накладываются крайне жесткие ограничения на доступные ресурсы, включая оперативную память, габариты процессора и потребляемую мощность.

13.2. Характеристики встроенных операционных систем

Управлять простой встроенной системой, обладающей несложной функциональностью, можно из одной специализированной программы или ряда программ, не прибегая к услугам другого программного обеспечения. Как правило, более сложные встроенные системы включают операционную систему. И хотя для встроенной системы в принципе можно воспользоваться универсальной операционной системой (например, Linux), ограничения, накладываемые на объем оперативной памяти и потребляемую мощность, а также требования к режиму работы в реальном времени обычно предписывают пользоваться специализированной операционной системой, предназначенной для применения в окружении конкретной встроенной системы.

Ниже перечислен ряд особых характеристик и требований к проектированию встроенных операционных систем.

- **Работа в реальном времени.** Во многих встроенных системах правильность вычисления отчасти зависит от времени получения его результата. Зачастую ограничения реального времени диктуются требованиями к внешнему интерфейсу и устойчивости управления.
- **Реагирующее действие.** Встроенное программное обеспечение может выполняться в ответ на внешние события. Если эти события не наступают периодически или через прогнозируемые промежутки времени, встроенное программное обеспечение, возможно, должно принять во внимание наихудшие условия и задать приоритеты выполнения процедур.
- **Конфигурируемость.** Вследствие большого разнообразия встроенных систем к функциональным возможностям встроенной операционной системы предъявляются самые разные (как количественные, так и качественные) требования. Следовательно, встраиваемая операционная система, предназначенная для применения в разнообразных встроенных системах, должна допускать гибкое конфигурирование, чтобы предоставлять лишь те функциональные возможности, которые требуются для конкретного приложения или комплекта оборудования. Характерные тому примеры приведены в [164]: применение функций компоновки и загрузки для выбора только тех модулей операционной системы, которые требуется загрузить; условная компиляция; определение соответствующих подклассов, если применяется объектно-ориентированная структура. Но при проектировании встроенных систем с большим количеством производных специально настраиваемых операционных систем могут возникнуть трудности, связанные с верификацией. На подобную потенциальную трудность для eCos указывает Такада в [250].
- **Гибкость устройств ввода-вывода.** Практически не существует такого устройства ввода-вывода, которое требовалось бы поддерживать во всех версиях операционных систем, и имеется большое разнообразие таких устройств. Поэтому в [164] предлагается поддержка относительно медленных устройств ввода-вывода (например, жестких дисков и сетевых интерфейсов) с помощью специальных задач вместо того, чтобы интегрировать их драйверы в ядро операционной системы.

- **Рационализированные механизмы защиты.** Как правило, встроенные системы предназначены для выполнения ограниченных, вполне определенных функций. Непротестированные программы редко внедряются в программное обеспечение. Следовательно, после настройки и тестирования программное обеспечение можно рассматривать как надежное. Таким образом, кроме мер безопасности, у встроенных систем могут быть ограниченные механизмы защиты. Например, команды ввода-вывода не обязаны быть привилегированными, чтобы операционная система могла перехватывать управление; задачи могут самостоятельно выполнять свой ввод-вывод. Аналогично могут быть сокращены до минимума механизмы защиты памяти. В [164] приведен следующий тому пример: пусть `switch` соответствует отображаемому в память адресу ввода-вывода значения, которое требуется проверять в ходе операции ввода-вывода. Мы можем позволить программе ввода-вывода выполнить отдельную команду наподобие загрузки этого `switch` в регистр для определения текущего значения. Такой подход предпочтителен, чем использование вызова служб операционной системы, которые приводят к накладным расходам на сохранение и восстановление контекста задачи.
- **Непосредственное применение прерываний.** В универсальных операционных системах любому пользовательскому процессу обычно не разрешается пользоваться прерываниями непосредственно. В [164] перечислены причины, по которым прерывания могут непосредственно запускать и останавливать задачи (например, сохраняя адрес запуска задачи в таблице адресов векторов прерываний) вместо того, чтобы проходить обычные для операционных систем процедуры обслуживания прерываний: 1) встроенные системы можно рассматривать как тщательно проверенные, с весьма редкими модификациями операционной системы или прикладного кода; 2) защита не является обязательной, как пояснялось в предыдущем абзаце; 3) требуется эффективное управление самыми разнообразными устройствами.

Исходные и целевые среды

Главное отличие настольных и серверных дистрибутивов операционной системы Linux от встроенных заключается в том, что настольное и серверное программное обеспечение, как правило, компилируется или конфигурируется на той платформе, на которой оно будет выполняться, тогда как встроенные дистрибутивы Linux обычно компилируются или конфигурируются на одной платформе, называемой исходной (host), а предназначаются для выполнения на другой платформе, называемой целевой (target, рис. 13.4). Основными компонентами, которые сначала разрабатываются на исходной платформе, а затем переносятся в целевую систему, являются начальный загрузчик, ядро и корневая файловая система.

Начальный загрузчик

Это небольшая программа, загружающая операционную систему в оперативную память после включения электропитания. Она отвечает за процесс первоначальной загрузки системы и загрузку ее ядра в основную память.

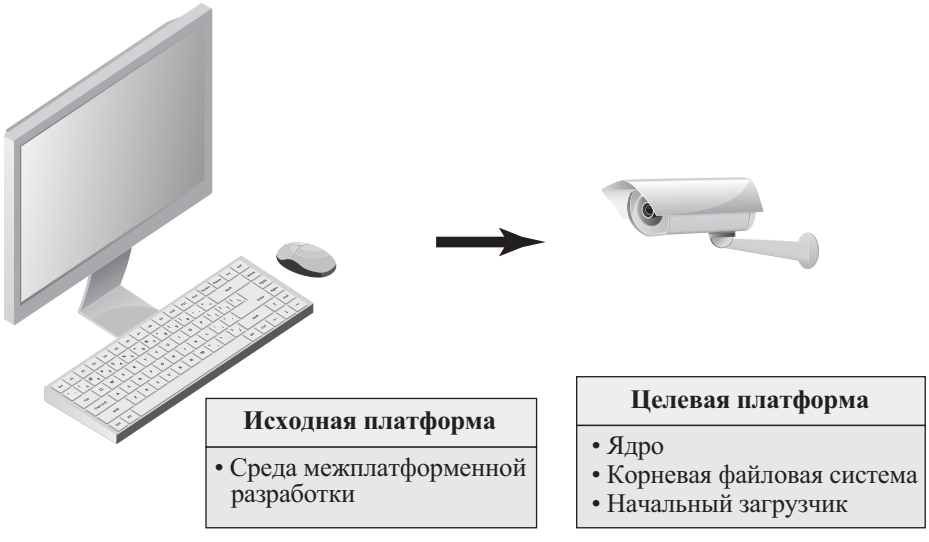

Рис. 13.4. Исходная и целевая платформы

Ниже приведена типичная последовательность начальной загрузки во встроенной системе.

1. Процессор во встроенной системе выполняет код, хранящийся в ПЗУ, чтобы загрузить начальный загрузчик первой стадии из внутренней флеш-памяти, SD-карты памяти или порта последовательного ввода-вывода.
2. Начальный загрузчик первой стадии инициализирует контроллер памяти и несколько периферийных устройств, а также загружает начальный загрузчик второй стадии в ОЗУ. Никакое взаимодействие с ним невозможно; обычно этот начальный загрузчик предоставляется поставщиком процессора в ПЗУ.
3. Начальный загрузчик второй стадии загружает ядро и корневую файловую систему из флеш-памяти в основную память (ОЗУ). Ядро и корневая файловая система обычно хранятся во флеш-памяти в виде упакованных файлов, так что процесс начальной загрузки отчасти состоит в распаковке файлов в бинарные образы ядра и корневой файловой системы. Затем начальный загрузчик передает управление ядру. Как правило, на второй стадии данного процесса применяется начальный загрузчик с открытым кодом.

Ядро

Полное ядро включает в себя целый ряд отдельных модулей, выполняющих среди прочего следующие функции.

- Управление памятью
- Управление процессами и потоками выполнения
- Межпроцессное взаимодействие; таймеры
- Ввод-вывод, поддержка сети, звука, хранения данных, графики и прочего с помощью драйверов
- Организация различных файловых систем

- Организация сети
- Управление электропитанием

Из программного обеспечения полного ядра данной операционной системы для встроенной системы опускается целый ряд дополнительных компонентов. Так, если аппаратные средства встроенной системы не поддерживают страничный обмен, то подсистему управления памятью можно удалить. Полное ядро будет включать в себя несколько файловых систем, драйверов устройств и так далее, из которых могут потребоваться лишь некоторые.

Как упоминалось ранее, главное отличие настольных и серверных дистрибутивов Linux от встроенных заключается в том, что настольное и серверное программное обеспечение, как правило, компилируется на той платформе, на которой оно будет выполняться, тогда как встраиваемые дистрибутивы Linux обычно компилируются на одной платформе, а предназначаются для выполнения на другой. Программное обеспечение, предназначенное для этой цели, называется *кросскомпилятором* (*межплатформенным компилятором*). Его применение наглядно показано на рис. 13.5.

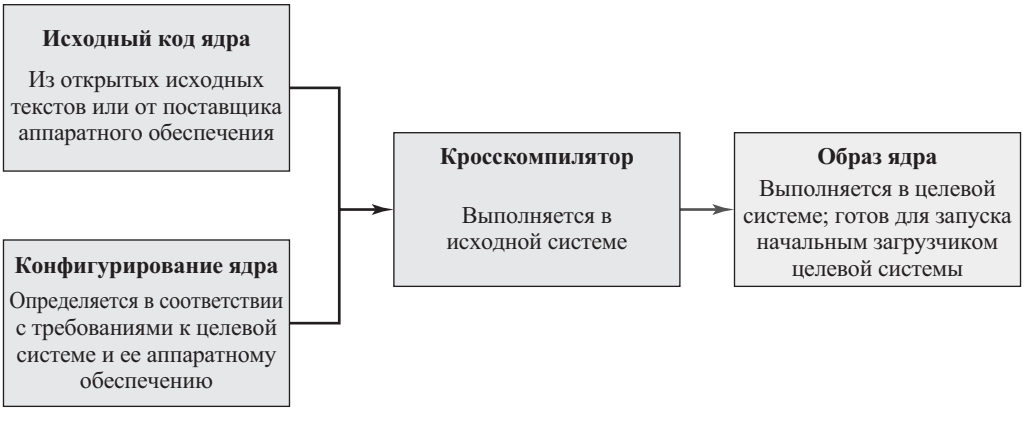

Рис. 13.5. Компиляция ядра

Корневая файловая система

Во встроенной или любой другой операционной системе имеется единая глобальная иерархия каталогов и файлов, предназначенная для представления всех файлов в системе. На вершине такой иерархии (в ее корне) находится корневая файловая система, в которой содержатся все файлы, требующиеся для нормальной работы системы. Корневая файловая система встроенной операционной системы подобна используемой на рабочей станции или сервере, с тем отличием, что она содержит лишь минимальный набор приложений, библиотек и файлов, требующихся для функционирования системы.

Подходы к разработке

Имеются два общих подхода к разработке встроенной операционной системы. Первый подход состоит в том, чтобы взять существующую операционную систему и приспособить ее для применения во встроенном варианте. Другой подход состоит в разработке и реализации операционной системы, предназначенной исключительно для встроенного применения.

Адаптация существующей коммерческой операционной системы

Существующая коммерческая операционная система может быть использована для разработки встроенной системы; для этого необходимо дополнить ее возможностями функционирования в реальном времени, рационализировать ее работу и добавить необходимые функциональные средства. При таком подходе к разработке встроенной системы обычно применяется не только Linux, но и FreeBSD, Windows и другие универсальные операционные системы, хотя зачастую они работают медленнее и менее предсказуемо, чем специализированные встроенные операционные системы. Преимущество такого подхода заключается в том, что встроенная операционная система, производная от коммерческой универсальной операционной системы, основывается на ряде знакомых интерфейсов, что упрощает ее переносимость.

Недостаток же применения универсальной операционной системы для разработки встроенной заключается в том, что она не оптимизирована для применения в реальном времени и во встроенном варианте. Следовательно, для достижения необходимой производительности может потребоваться значительная ее модификация. В частности, типичная операционная система оптимизируется для среднего, а не наихудшего случая планирования заданий, обычно назначая ресурсы по требованию и пренебрегая большинством, если не всеми семантическими сведениями о приложении.

Специально разработанная встроенная операционная система

Значительное количество операционных систем разработаны с самого начала для применения во встроенном варианте. Двумя характерными примерами такого подхода к разработке встроенных систем являются операционные системы eCos и TinyOS, обсуждаемые далее в этой главе.

Ниже перечислены типичные характеристики специализированной встроенной операционной системы.

- Наличие переключателя быстрых и упрощенных процессов или потоков исполнения.
- Стратегия планирования, реализуемая в виде модуля диспетчеризации в реальном времени как часть планировщика, а не отдельного компонента.
- Малые размеры.
- Быстрое реагирование на внешние прерывания. Типичным требованием к встроенной операционной системе является время отклика менее 10 мс.
- Минимизация промежутков времени, в течение которых запрещены прерывания.
- Предоставление для управления памятью разделов фиксированного или переменного размера, а также возможности блокировать код и данные в памяти.
- Предоставление специальных последовательных файлов, в которых можно накапливать данные с большой скоростью.

Чтобы каким-то образом удовлетворять временны*м* ограничениям, ядро

- предоставляет ограниченное время для выполнения большинства примитивов;
- поддерживает часы реального времени;
- выдает специальные предупреждающие сигналы и блокировки по времени;
- поддерживает правила организации очередей (например, первоочередное обслуживание запросов с самым ранним сроком выполнения), а также примитивов для сжатия сообщения в начале очереди;
- предоставляет примитивы для задержки обработки на фиксированное время, а также для приостановки и возобновления процесса выполнения.

Перечисленные выше характеристики имеют немало общего с требованиями к функционированию встроенных операционных систем в реальном времени. Тем не менее в требованиях к сложным встроенным системам основной акцент может быть сделан на предсказуемое, а не на быстрое действие, а для этого придется принимать совсем другие проектные решения, особенно в области планирования заданий.

13.3. Встроенная система Linux

Термин *встроенная система Linux* просто означает версию Linux, выполняемую во встроенной системе. Как правило, во встроенной системе Linux применяется один из официальных выпусков ядра, несмотря на то что в некоторых системах используется модифицированное ядро, приспособленное к конкретной конфигурации оборудования или для поддержки определенного класса приложений. Прежде всего, ядро встроенной системы Linux отличается от ядра обычной операционной системы Linux, установленной на рабочей станции или сервере, конфигурацией сборки и средой разработки.

В этом разделе сначала выделяются некоторые из основных отличий встроенной Linux от ее версии, работающей на настольном компьютере или сервере, а затем описывается распространенное программное обеспечение под названием “μClinux”.

Характеристики встроенной системы Linux

Размер ядра

Настольные и серверные системы Linux должны поддерживать большое число устройств, поскольку в Linux применяются самые разные конфигурации. Такие системы должны поддерживать и целый ряд протоколов передачи и обмена данными, а следовательно, их можно применять для самых разных целей. Встроенным устройствам, как правило, требуется поддержка определенного ряда конкретных устройств, периферийных устройств и протоколов в зависимости от установленного оборудования и предназначения. К счастью, ядро Linux в высшей степени поддается конфигурированию в плане архитектуры, для которой оно скомпилировано, а также поддерживаемых в нем процессоров и устройств.

Дистрибутив встроенной системы Linux является версией Linux, специально приспособленной под ограничения, накладываемые на размер и оборудование встроенных устройств. В его состав входят программные пакеты, поддерживающие различные службы и приложения, действующие на подобных устройствах. Таким образом, ядро встроенной системы Linux оказывается намного меньше, чем ядро обычной системы Linux.

Объем памяти

Размер встроенной системы Linux классифицируется в [76] по объему доступного ПЗУ или ОЗУ в трех обширных категориях малых, средних и крупных систем. В частности, малые системы характеризуются наличием маломощного процессора, ПЗУ объемом минимум 2 Мбайт и ОЗУ объемом 4 Мбайт. Средние по размеру системы отличаются наличием среднемощного процессора, ПЗУ объемом около 32 Мбайт и ОЗУ объемом 64 Мбайт. А крупные системы характеризуются наличием мощного процессора или даже нескольких процессоров и большими объемами как оперативной, так и постоянной памяти.

В системе без постоянной памяти все ядро Linux должно вмещаться в ОЗУ и ПЗУ, чего нельзя сказать о полномасштабной современной системе Linux. В качестве иллюстрации данного положения на рис. 13.6 показано увеличение размера ядра полномасштабной системы Linux со временем. Безусловно, любую систему Linux можно сконфигурировать лишь с некоторыми компонентами из полного выпуска. Тем не менее на диаграмме, приведенной на рис. 13.6, наглядно показано, что, в особенности для мелких и средних по размеру встроенных систем, могут быть опущены значительные объемы ядра.

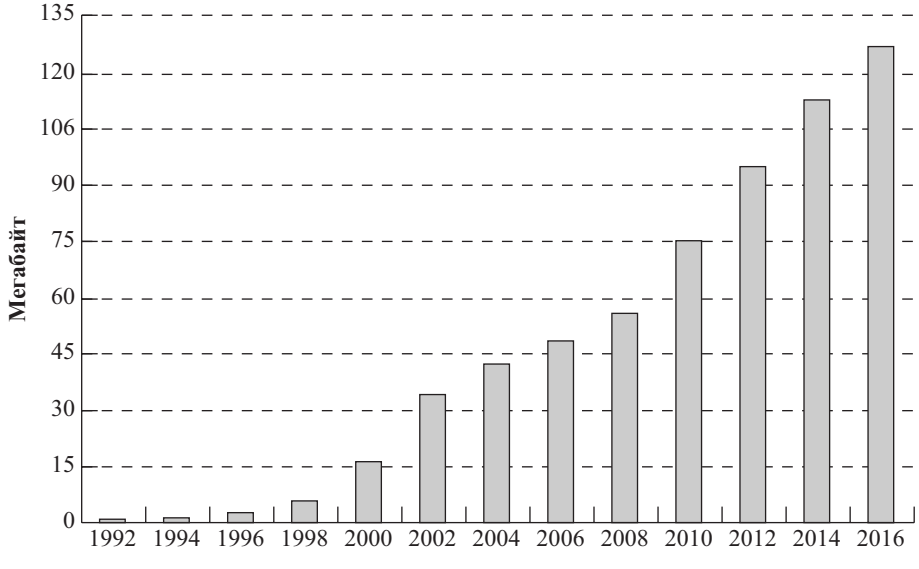

Рис. 13.6. Размер ядра Linux, показанный в сжатом формате файлов GZIP

Прочие характеристики

К другим характеристикам встроенных систем Linux относятся следующие.

- **Временны*е* ограничения.** Жесткие временны*е* ограничения требуют от системы реакции на внешние воздействия в течение указанного промежутка времени. А мягкие временны*е* ограничения пригодны для тех систем, в которых медленная реакция системы является некритичной.

- **Подключаемость к сети.** Означает способность системы работать в сети. Буквально все современные встроенные устройства обладают такой способностью, подключаясь, как правило, к беспроводной сети.
- **Степень взаимодействия с пользователем.** Одни устройства сосредоточены на взаимодействии с пользователем, тогда как другие, в том числе управляющие промышленными процессами, могут предоставлять для взаимодействия с пользователем очень простой интерфейс (например, через светодиоды и кнопки). Имеются и такие устройства, у которых взаимодействие с конечным пользователем отсутствует (например, датчики, действующие в Интернете вещей, собирающие данные и передающие их в облако).

В табл. 13.1, взятой из [76], приведены характеристики некоторых коммерчески доступных встроенных систем, в которых применяется ядро Linux.

Таблица 13.1. Характеристики примеров встроенных систем Linux

Описание	Тип	Размер	Временные ограничения	Подключаемость к сети	Степень взаимодействия с пользователем
Устройства управления акселератором	Управление промышленными процессами	Средний	Жесткие	Да	Низкая
Автоматизированная обучающая система	Авиакосмическая отрасль	Крупный	Жесткие	Нет	Высокая
Устройство типа Bluetooth для доступа к местным данным	Организация сети	Мелкий	Мягкие	Да	Очень низкая
Преобразователь протоколов для управления системой сбора данных	Управление промышленными процессами	Средний	Жесткие	Нет	Очень низкая
Карманный персональный компьютер	Бытовая электроника	Средний	Мягкие	Да	Очень высокая
Устройство управления двигателем, применяемое в системе управления космическим аппаратом	Авиакосмическая отрасль	Крупный	Жесткие	Да	Высокая

Файловые системы встроенного Linux

В некоторых приложениях могут создаваться относительно небольшие файловые системы, которые предназначены для применения только во время работы самого приложения и могут храниться в основной памяти. Но в целом файловая система должна храниться в долговременной памяти (например, во флеш-памяти или на традиционных

дисковых запоминающих устройствах). Впрочем, для большинства встроенных систем внутренний или внешний диск не подходит, и поэтому в качестве долговременной памяти обычно применяется флеш-память.

Что же касается остальных свойств встроенной системы Linux, то ее файловая система должна быть как можно более компактной. Для применения во встроенных системах был разработан целый ряд компактных файловых систем. В качестве примера ниже перечислены некоторые из наиболее употребительных систем данной категории.

- **cramfs.** Упакованная файловая система в ОЗУ (Compressed RAM File System), простая и доступная только для чтения. Она служит для того, чтобы минимизировать размер, максимально эффективно используя емкость исходного запоминающего устройства. В файловых системах cramfs отдельные файлы упаковываются в блоки, совпадающие по размеру со страницами в Linux (как правило, 4096 байт или 4 Мбайт в зависимости от версии ядра и конфигурации), чтобы обеспечить эффективный произвольный доступ к содержимому файлов.
- **squashfs.** Как и cramfs, squashfs является упакованной доступной только для чтения файловой системой, предназначенной для применения в средах с малой или ограниченной емкостью запоминающего устройства (например, во встроенных системах Linux).
- **jffs2.** Вторая версия файловой системы для флеш-памяти с журналированием (Journaling Flash File System). Она предназначена для применения в устройствах флеш-памяти типа NOR и NAND с особым акцентом на такие вопросы эксплуатации флеш-памяти, как выравнивание износа.
- **ubifs.** Файловая система с несортированными образами блоков (Unsorted Block Image File System), которая обычно обеспечивает лучшую производительность, чем файловая система jffs2 в крупных устройствах флеш-памяти. Для дополнительного повышения производительности в ней также поддерживается кеширование записи.
- **yaffs2.** Еще одна версия файловой системы (Yet another Flash File System) для флеш-памяти, обеспечивающая быстрое и надежное хранение файлов во флеш-памяти. Для хранения данных состояния файловой системы yaffs2 требуется меньше места в ОЗУ, чем в таких файловых системах, как jffs2. И, как правило, она обеспечивает лучшую производительность, если запись данных в файловой системе выполняется слишком часто.

Преимущества встроенных систем Linux

Встроенные версии Linux стали появляться еще в 1999 году. В целом ряде компаний были разработаны свои версии, предназначенные для конкретных платформ. Ниже перечислены преимущества применения Linux в качестве основания для разработки встроенной операционной системы.

- **Независимость поставщиков.** Поставщик платформы не зависит от поставщика конкретной встроенной системы, чтобы предоставить необходимые функциональные средства и уложиться в крайние сроки развертывания.

- **Разнообразная аппаратная поддержка.** В системе Linux поддерживается обширный ряд архитектур процессоров и периферийных устройств, и благодаря этому она оказывается вполне пригодной для разработки буквально каждой встроенной системы.
- **Малые затраты.** Применение Linux позволяет свести к минимуму затраты на разработку и обучение.
- **Открытость исходного кода.** Применение Linux дает все преимущества программного обеспечения с открытым исходным кодом.

μClinux

μClinux (т.е. микроконтроллерная Linux) является весьма распространенной разновидностью ядра Linux с открытым исходным кодом, предназначенного для микроконтроллеров и прочих очень малых встроенных систем. В силу модульного характера Linux совсем не трудно сократить операционную среду, исключив из нее служебные программы, инструментальные средства и прочие системные службы, которые не нужны во встроенной среде. В этом, собственно, состоит основной принцип проектирования μClinux.

Чтобы дать ясное представление о размере загружаемого образа μClinux (ядра и корневой файловой системы), обратимся к опыту компании EmCraft Systems, разрабатывающей системы на уровне печатных плат с помощью микроконтроллеров Cortex-M и микропроцессоров Cortex-A [74]. Это едва ли не самые малые встроенные системы, в которых применяется μClinux. Так, минимальная конфигурация μClinux может занимать не больше 0,5 Мбайт, хотя поставщик посчитал вполне практичным размер загружаемого образа, включая Ethernet, TCP/IP, приемлемый набор инструментальных средств из пользовательского пространства и сконфигурированных приложений, в пределах от 1,5 до 2 Мбайт. А объем памяти, требующийся для μClinux во время выполнения, должен находиться в пределах от 8 до 32 Мбайт. Эти числовые показатели значительно меньше, чем у типичной системы Linux.

Сравнение с полномасштабной системой Linux

Главные отличия μClinux от Linux для крупных систем (подробнее об этом см. в [165]) заключаются в следующем.

- Linux является многопользовательской операционной системой, основанной на UNIX, а μClinux — версией Linux, предназначенной для разработки встроенных систем (как правило, без взаимодействия с пользователем).
- В отличие от Linux, в μClinux не поддерживается управление памятью. Следовательно, виртуальные адресные пространства в μClinux не требуются, а приложения должны быть привязаны к абсолютным адресам.
- В ядре Linux поддерживается отдельное пространство виртуальных адресов для каждого процесса, а в μClinux — единое адресное пространство, общее для всех процессов.
- Адресное пространство в Linux восстанавливается при переключении контекста, чего нельзя сказать о μClinux.

- В отличие от Linux, в μClinux не обеспечивается системный вызов `fork()`, а вместо этого применяется функция `vfork()`. По существу, системный вызов с помощью функции `fork()` приводит к созданию дубликата, во многом похожего на вызывающий процесс (однако при этом копируется не все (например ресурсы в некоторых реализациях ограничены), хотя основной замысел состоит в том, чтобы создать как можно более полную копию). Новый (порожденный) процесс получает другой идентификатор процесса и имеет тот же идентификатор родительского процесса. Главное отличие функции `vfork()` от функции `fork()` заключается в следующем: когда новый процесс создается с помощью функции `vfork()`, родительский процесс временно приостанавливается, а порожденный процесс может позаимствовать адресное пространство своего родителя. И так продолжается до тех пор, пока завершится порожденный процесс или же им будет вызвана функция `execve()`, после чего родительский процесс продолжится.
- μClinux модифицирует драйверы устройств для использования локальной системной шины вместо ISA или PCI.

Самое значительное отличие полномасштабной системы Linux от μClinux относится к области управления памятью. Отсутствие поддержки управления памятью в μClinux может иметь целый ряд последствий, в том числе следующие.

- Основная память, выделяемая для процесса, в общем случае должна быть непрерывной. Если же выполняется свопинг целого ряда процессов, это может привести к фрагментации памяти (см. рис. 7.4). Тем не менее во встроенных системах, как правило, имеется фиксированное множество процессов, загружаемых во время начальной загрузки и работающих до следующего сброса. Следовательно, такая функциональная возможность обычно не требуется.
- μClinux не может расширить память для работающих процессов, поскольку с ней могут быть смежными другие процессы. Следовательно, вызовы функций `brk()` и `sbrk()`, динамически изменяющих объем используемого пространства, выделяемого для сегмента данных из вызывающего процесса, недоступны. Но в μClinux все же предоставлена реализация функции `malloc()`, предназначенная для выделения блока памяти из глобального пула памяти.
- В μClinux отсутствует стек динамических приложений. Это может привести к переполнению стека, а значит, и к повреждению памяти. Во избежание этого на стадии разработки и конфигурирования приложений необходимо принять соответствующие меры.
- В μClinux не обеспечивается защита памяти, что представляет риск повреждения одним приложением части другого приложения или даже ядра. Впрочем, этот недостаток все же исправлен в некоторых реализациях. Например, архитектура Cortex-M3/M4 предоставляет механизм защиты памяти под названием “MPU” (Memory Protection Unit — блок защиты памяти). Используя механизм MPU, компания Emcraft Systems внедрила в ядро дополнительное функциональное средство, реализующее взаимную защиту как самих процессов, так и ядра и процессов, сопоставимую с механизмами защиты памяти, реализованными в Linux с помощью блока управления памятью (MMU) [130].

µClibc

Это системная библиотека на C, первоначально разработанная для поддержки µClinux и обычно применяемая вместе с µClinux, хотя библиотекой µClibc можно пользоваться и вместе с другими ядрами Linux. Основное назначение µClibc — представить системную библиотеку, написанную на языке C и пригодную для разработки встроенных систем Linux. Она намного меньше библиотеки GNU C (glibc), широко применяемой в системах Linux, хотя практически все приложения, поддерживаемые в библиотеке glibc, идеально взаимодействуют и с µClibc. Перенос приложений из библиотеки glibc в библиотеку µClibc, как правило, предусматривает лишь перекомпиляцию исходного кода. В библиотеке µClibc поддерживаются также совместно используемые библиотеки и поточная обработка.

В табл. 13.2, составленной по материалам, взятым из [4], сравниваются размеры функций в обеих упомянутых выше библиотеках. Как видите, экономия занимаемого места вполне очевидна. Такая экономия достигается благодаря запрету некоторых функциональных средств по умолчанию и радикальной переделке исходного кода с целью исключить его избыточность. А на рис. 13.7 показана высокоуровневая программная архитектура встроенной системы, построенной с помощью µClinux и µClibc.

Таблица 13.2. Размеры некоторых функций из библиотек µClibc и glibc

Имя функции из glibc	Размер функции glibc, Кбайт	Имя функции из µClibc	Размер функции µClibc, Кбайт
`libc-2.3.2.so`	1200	`libuClibc-0.9.2.7.so`	284
`ld-2.3.2.so`	92	`libcrypt-0.9.2.7.so`	20
`libcrypt-2.3.2.so`	20	`libdl-0.9.2.7.so`	12
`libdl-2.3.2.so`	12	`libm-0.9.2.7.so`	8
`libm-2.3.2.so`	136	`libnsl-0.9.2.7so`	56
`libnsl-2.3.2.so`	76	`libpthread-0.9.2.7.so`	4
`libpthread-2.3.2.so`	84	`libresolv-0.9.2.7.so`	84
`libresolv-2.3.2.so`	68	`libutil-0.9.2.7.so`	4
`libutil-2.3.2.so`	8	`libcrypt-0.9.2.7.so`	8

Android

Как подробно обсуждалось ранее в этой книге, Android является встроенной операционной системой, основанной на ядре Linux. Следовательно, Android можно с полным основанием рассматривать как пример встроенной системы Linux. Тем не менее многие разработчики встроенных систем Linux не считают Android примером системы данной категории [47]. С их точки зрения, у классического встроенного устройства имеется фиксированный набор функций, прошитых изготовителем. Android же в большей степени относится к категории платформенных операционных систем, поддерживающих разнообразные приложения, различающиеся на разных платформах. Кроме того, Android является вертикально интегрированной системой, включая модификации ядра Linux, специально сделанные для Android.

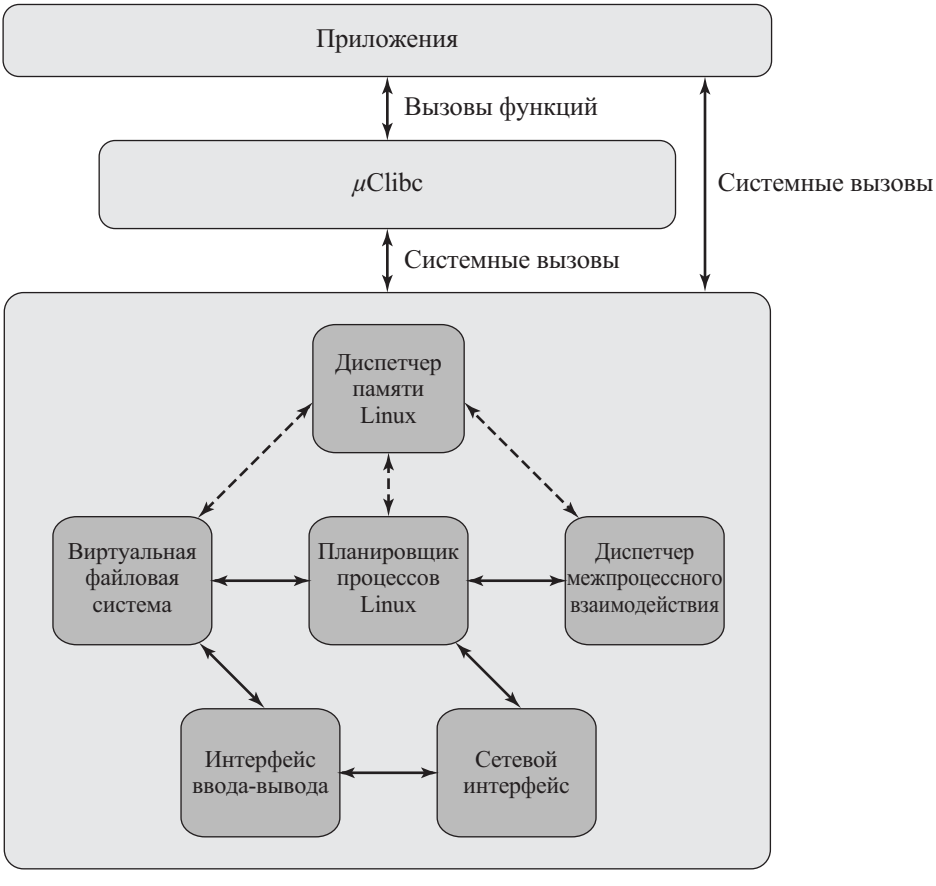

Рис. 13.7. Программная архитектура встроенной системы, построенной на основе *μClinux и μClibc*

Основное внимание в Android уделяется вертикальной интеграции ядра Linux и компонентов из пользовательского пространства Android. В конечном счете это вопрос семантики, а "официальное" определение Android как встроенной системы Linux, на которое можно было бы положиться, отсутствует.

13.4. TinyOS

В системе TinyOS представлен более рациональный подход к разработке встроенных операционных систем, чем основанный на коммерческих универсальных операционных системах (например, встроенной версии Linux). Следовательно, TinyOS и аналогичные ей системы в большей степени пригодны для разработки небольших встроенных систем с жесткими требованиями к памяти, времени обработки и реагирования, потребляемой мощности и т.д. Процесс рационализации заходит в TinyOS довольно далеко, а в результате получается самая маленькая операционная система для встроенных систем. Базовой операционной системе требуется всего 400 байт памяти для кода и данных, вместе взятых.