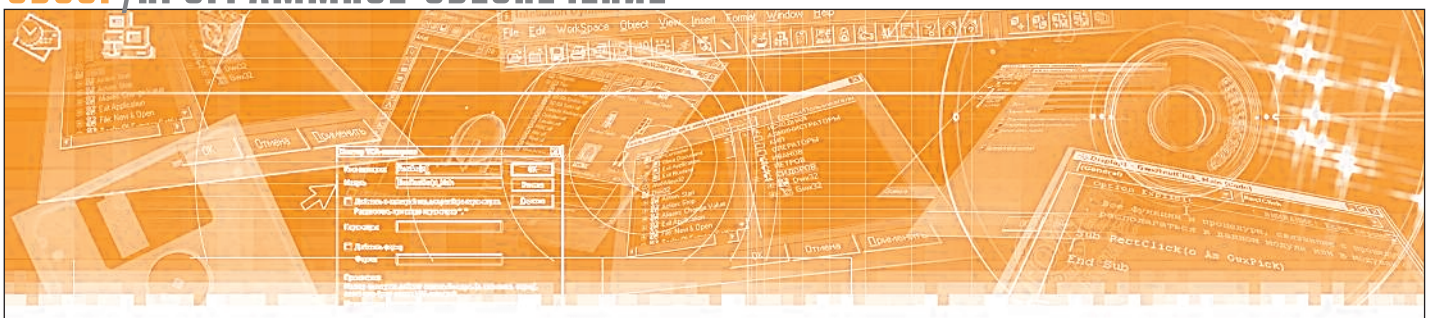

Гленн Сайлер

Операционные системы VxWorks и Wind River Linux: подходы к реализации реального времени

В статье рассматривается классификация задач реального времени, позволяющая принимать решение о выборе операционных систем для их реализации. Представлены два диаметрально противоположных подхода к реализации реального времени в ОС: добавление функционала реального времени в ОС общего назначения и, наоборот, расширение функциональности ОС реального времени в прикладную область. В качестве примера приводятся встраиваемые операционные системы Wind River Linux и Wind River VxWorks.

Введение

В портфель программных продуктов для встраиваемых приложений, разрабатываемых и поставляемых компанией Wind River, входят две встраиваемые операционные системы реального времени: зарекомендовавшая себя VxWorks и относительно молодая Wind River Linux. Учитывая постоянно растущее число продуктов в каждой рыночной «экологической нише», было бы полезно взвесить все за и против использования VxWorks и Linux в различных приложениях. В данной статье, в отличие от общепринятой трактовки в контексте характеристик операционных систем (ОС), понятие «реальное время» переопределяется с точки зрения требований конечного приложения. Статья разъясняет разницу между требованиями условного (conditional) и гарантированного (guaranteed) реального времени, приводит примеры приложений, к которым предъявляются требования обоих типов, а также иллюстрирует, как в VxWorks и Linux достигается необходимый уровень детерминизма.

Условное и гарантированное реальное время

Термин «реальное время» может поразному трактоваться в зависимости от задачи. Каким-то приложениям требуется не более чем усреднённый отклик на некотором временном интервале, а каким-то – чтобы время реакции никогда не превышало установленный лимит.

Самый простой способ определить реальное время в контексте прикладной задачи – это разделить его на условное и гарантированное. Несмотря на то что точные определения этих категорий могут сильно варьироваться, приведём ряд общих соображений.

- **Условное реальное время** подразумевает гарантию предоставления определённого процента вычислительной мощности (то есть процессорного времени) на некотором временном интервале. Пример приложения, требующего поддержки условного реального времени, – программа, которой необходимо 10% процессорного времени на протяжении каждых 100 мс. Иными словами, для обеспечения корректного функционирования ей необходимо раз в каждые 100 мс выполняться в течение 10 мс, а как именно она получит эти 10 мс – это уже не так важно.
- **Гарантированное реальное время**, в отличие от условного, завязано не на процент вычислительной мощности, а на время реакции; иными словами, приложению для корректного функционирования необходимо реагировать на внешние асинхронные события за чётко регламентированное время. В качестве примера можно привести обработку периодически возникающих прерываний, где максимальное время обработки прерываний не должно превышать период их поступления, чтобы ни одного прерывания не потерялось. Для приложений, требующих гарантированного реального времени, часто характерна высокая стоимость отказа, например, несвоевременная реакция на событие в управляющем приложении (скажем, антиблокировочной системе транспортного средства) может привести к катастрофическим последствиям.

(Автор не случайно вводит дополнительную классификацию реального времени на условное и гарантированное. Дело в том, что похожее по смыслу устоявшееся разделение реального времени на жёсткое и мягкое на самом деле относится к свойствам самих ОС, а не к требованиям конечной задачи. Например, задача, требующая в рамках приведённой автором классификации реального времени условного, может быть реализована с применением ОС как жёсткого, так и мягкого реального времени, а то и без ОС вообще. Таким образом, в нарушении принципа «бритвы Оккама» автора обвинить нельзя. – *Прим. пер.*)

Обеспечение гарантированного реального времени – узкоспециализированная и нетривиальная задача, и в ряде приложений это требование можно смягчить до реального времени условного. Самый простой пример – воспроизведение звука. Чтобы корректно воспроизводить звук при широко распространённой частоте квантования 44,1 кГц, приложению необходимо передавать оборудованию 32 бита данных каждые 22,7 мкс. Поскольку все остальные сервисы ОС могут иметь невытесняемые сегменты кода, выполняющиеся более чем 22,7 мкс, задержка может превысить необходимую величину; чтобы этого не произошло, применяется аппаратная буферизация, благодаря которой задержки, вносимые вытеснением, сглаживаются. Использование 32-разрядного буфера на 4096 элементов смягчает требование приложения к времени реакции системы до необходимости заполнения буфера ориентировочно раз в 100 мс.

Примеры приложений гарантированного реального времени

Для многих приложений характерны жёсткие требования к реальному времени, которые не могут быть сведены к реальному времени условному. Приведём примеры областей, в которых приложения часто требуют гарантированного реального времени.

- **Авиация, космонавтика и ВПК.** Современные аэрокосмические и военные приложения включают в себя авионику нового поколения, пункты управления, системы навигации и захвата/сопровождения цели, управления оружием и т.п. Устройства, предназначенные для использования в оборонных системах, гражданской авиации и космонавтике, предъявляют одни из самых жёстких требований к ПО и требуют применения надёжной, высокопроизводительной и «коммуникабельной» (в оригинале “connected”. – *Прим. пер.*) операционной системы.
- **АСУ ТП, приборостроение и робототехника.** Промышленные приложения требуют одновременно высокой надёжности, предсказуемости и производительности. Промышленное производство, энергетика, транспорт – все эти отрасли по определению предъявляют к приложениям повышенные требования по функциональной безопасности, поскольку отказы могут стоить миллионы долларов, а то и человеческих жизней. Будь разрабатываемое устройство хоть манипулятором сборочного конвейера, хоть удалённо управляемым колёсным роботом, требования к предсказуемости и производительности ПО будут одинаково жёсткими.
- **Телекоммуникации.** Современные приложения передачи данных и голоса предъявляют одновременно широкий спектр требований условного и гарантированного реального времени наряду с требованиями высокой пропускной способности. Потоковая передача голоса и других медиаданных требует качества обслуживания, которое выходит за пределы возможностей ОС общего назначения. Высокая плотность сетевого трафика также может вызвать перегрузку стеков протоколов, поставляемых в составе универсальных ОС. Типичный пример такой задачи – это маршрутизация, требующая управления потоками из тысяч IP-пакетов в реальном времени.
- **Мобильная связь.** Реальное время – одно из ключевых требований к современным мобильным телефонам на базе одноядерных процессоров. Применение одноядерных процессоров позволяет производителям мобильных телефонов снизить себестоимость; однако это одновременно требует реализации на одном и том же ядре и основного протокола связи с его жёсткими требованиями детерминизма, и пользовательских приложений реального времени, включая потоковое мультимедиа.

Предсказуемость реакции: требования и способы измерения

У всех описанных в предыдущем разделе приложений есть одна общая черта – необходимость в чётко регламентированных (на уровне единиц микросекунд) временах реакции на прерывание и перепланирования задач. Рассмотрим эти параметры подробнее.

Время реакции на прерывание

Время отклика системы часто измеряется как время её реакции на прерывание. Время *реакции на прерывание* складывается из длительности всех действий ядра ОС по инициированию его обработки, включая сохранение контекста задач, определение источника прерывания и передачу управления соответствующему обработчику.

В действительности время *реакции на прерывание* является сильно аппаратно-зависимым и определяется быстродействием оборудования; с программной стороны можно управлять только временем выполнения подпрограммы обработки прерывания (Interrupt Service Routine – ISR). Однако в большинстве приложений реального времени, перед тем как система сможет выполнить какие-то действия по факту возникновения прерывания, сначала должен быть поставлен на выполнение некий процесс, задача или поток. Время выполнения этой операции складывается из так называемых *времени перепланирования* и *времени переключения контекста*.

Время перепланирования и время переключения контекста

Время переключения контекста – это время, требуемое для передачи управления от одной задачи другой. Переключению контекста предшествует перепланирование, то есть принятие планировщиком ОС решения, какой задаче необходимо предоставить процессор; *время перепланирования* – это время, за которое планировщик данное решение принимает. (ОС VxWorks и Wind River Linux используют вытесняющий планировщик, поэтому решение о планировании задачи принимается на основании её приоритета, установленной дисциплины планирования и состояния соответствующих средств синхронизации. Например, задача высокого приоритета может быть забло-

Рис. 1. Реакция ОС на прерывание (с перепланированием)

кирована на семафоре; ISR изменяет значение семафора, что вызывает перепланирование и разблокирует задачу; поскольку эта задача имеет самый высокий приоритет среди готовых к выполнению, она получает процессор немедленно. Если при этом не включено карусельное (round-robin – RR) планирование, то задача будет выполняться, пока либо не заблокируется сама, либо не будет вытеснена более высокоприоритетной задачей. – *Прим. пер.*)

Все задержки, связанные с реакцией ОС на прерывание (в случае если в результате прерывания возникает перепланирование), показаны на рис. 1.

Существует множество способов обеспечить предсказуемость реакции ОС на внешнее событие. ОС реального времени (ОС РВ), такие как VxWorks, были созданы специально, чтобы удовлетворять этому требованию; Linux, с другой стороны, изначально разрабатывалась как операционная система общего назначения (General Purpose OS – GPOS).

Как превратить Linux в ОС реального времени?

За последние несколько лет был предпринят ряд усилий по улучшению производительности Linux, чтобы сделать её преимущества (например, широкий выбор прикладного ПО с открытым исходным текстом) доступными для приложений реального времени. В этом разделе статьи мы рассмотрим два базовых подхода к преобразованию Linux в ОС РВ.

PREEMPT_RT и обеспечение вытесняемости ядра

При использовании этого подхода ядро Linux модифицируется так, чтобы минимизировать время, проводимое им в невытесняемых секциях кода, когда оно не может реагировать на прерывания и соответствующие события перепланирования. Этот подход требует комплексной модификации ядра, чтобы охватить все невытесняемые секции, и поэтому представляет собой итерационный процесс, в потенциале требующий тестирования после внесения каждого изменения. Если обнаруживается код, время невытесняемого выполнения которого превышает установленный лимит, он перерабатывается, чтобы соответствовать заданным временным ограничениям.

В реальности процесс поиска слишком длинных невытесняемых секций является сугубо эмпирическим. Вычисление времени реакции для худшего случая (worst-case) требует исчерпывающего тестирования всех системных вызовов при всевозможных нагрузочных сценариях. На практике это, естественно, нереализуемо, и хотя этот метод и позволяет добиться приемлемых значений времени реакции, абсолютную гарантию их соблюдения дать невозможно.

Снижению времени реакции ядра Linux посвящён ряд открытых проектов, использующих различные подходы к обеспечению вытесняемости. Самый существенный из этих проектов – PREEMPT_RT – был начат Инго Молнаром (Ingo Molnar) и постепенно включается в основную ветвь кода ядра. Эта функциональность присутствует в составе платформ Wind River Linux.

Данный подход обладает рядом очевидных преимуществ; пожалуй, наиболее значимое из них то, что проект PREEMPT_RT широко признан как часть «официальной» Linux, поэтому его качество со временем улучшается (в частности, поддержка вытесняемости добавляется в код драйверов устройств). Однако есть и свои ограничения, поскольку архитектурно-зависимым участкам кода и драйверам, относящимся к недостаточно популярной аппаратуре, в сообществе может не уделяться необходимого внимания. В результате версии ядра Linux для различных процессорных архитектур и даже для различных плат могут иметь различную производительность. В большинстве случаев драйверы переносятся в модель PREEMPT_RT без изменений, но потенциальные проблемы с их производительностью и корректностью совместной работы классических и новых драйверов поднимают вопрос дополнительного тестирования.

Wind River Real-Time Core и концепция модуля реального времени (Real-Time Executive)

В данном варианте в системе одновременно присутствуют два ядра: ядро Linux и компактное ядро реального времени. Этот подход используется в Wind River Real-Time Core для Wind River Linux: там ядро Linux выполняется под управлением ядра реального времени как задача самого низкого приоритета и получает прерывания через слой виртуализации. Первичная обработка прерываний полностью лежит на ядре реального времени, и ядро Linux получает их только тогда, когда нет других готовых к выполнению задач; задачи же реального времени загружаются в пространство ядра и получают прерывания незамедлительно, что обеспечивает им скорость реакции, сравнимую с аппаратной.

Wind River Real-Time Core и пользовательские задачи Linux взаимодействуют посредством неблокирующих очередей и разделяемой памяти. С точки зрения прикладного программиста очереди выглядят наподобие стандартных для Linux символьных устройств, доступных через POSIX-

совместимые вызовы open/read/write/ioctl. Разделяемая память управляется через POSIX-совместимый вызов mmap.

Задачи реального времени реализуются в данном подходе как стандартные потоки POSIX, и их программирование не требует знаний об устройстве ядра Linux. Таким образом, разработка приложений реального времени, во-первых, значительно упрощается, а во-вторых, никак не влияет на происходящее в Linux-среде.

Сравнение Wind River Real-Time Core и PREEMPT_RT

Подход с использованием патча PREEMPT_RT несёт в себе возможность оставаться в рамках «официальной» Linux и хорошо подходит для ряда приложений реального времени типа обработки потоков аудио- и видеоинформации, но не способен обеспечить 100-процентную гарантию требуемой производительности.

Wind River Real-Time Core, с другой стороны, не входит в стандартную комплектацию Linux и не подпадает под условия лицензии General Public License (GPL), но зато предоставляет разработчику специализированную среду с гарантированными значениями времени реакции, одновременно позволяя выполнять приложения Linux без изменений.

ОС реального времени Wind River VxWorks

Оба описанных подхода подразумевают «изготовление» ОС реального времени из ОС общего назначения. Другой диаметрально противоположный подход заключается в том, чтобы добавить в ОС реального времени сервисы общего назначения. Этот подход используется в ОС VxWorks.

VxWorks – самая распространённая коммерческая ОС РВ в индустрии встраиваемых приложений. Она изначально разрабатывалась как ОС, которая обеспечивала бы быстрое и предсказуемое переключение контекста. Микроядро VxWorks поддерживает вытесняющее и карусельное планирование с 256 уровнями приоритета при неограниченном количестве задач. VxWorks определяет, какой задаче предоставить процессор, на основании дисциплины планирования; по умолчанию используется вытесняющее планирование, то есть задача с более высоким приоритетом всегда вытесняет менее приоритетную. В VxWorks для всех задач используется общее адресное пространство, что позволяет избежать затрат на трансляцию виртуальных адресов в физические.

Модель процесса реального времени (RTP)

Исторически в VxWorks присутствовали только задачи, выполняющиеся в контексте ядра, что позволяло получить выигрыш в производительности, скорости отработки системных вызовов и времени доступа к оборудованию, жертвуя при этом защитой компонентов ядра и абстракцией модели программирования. Такой компромисс требовал от разработчиков детальных знаний о работе с оборудованием и привносил дополнительный риск возникновения невосстанавливаемых отказов.

Начиная с версии 6.0 и старше, компания Wind River реализовала в VxWorks модель процесса реального времени (Real-Time Process – RTP), в рамках которой контексты ядра и пользователя разграничиваются. Модель RTP вводит возможность программирования в пользовательском кольце, полностью сохраняя и дополняя классический подход. Она также обеспечивает защиту памяти, не снижая при этом производительность традиционных приложений, работающих в пространстве ядра. Приложения, построенные в рамках модели RTP, выигрывают одновременно и от производительности ядра VxWorks, и от надёжности при работе с памятью и другими ресурсами.

Модель RTP также значительно расширяет набор средств, доступных прикладному программисту. Приложения, работающие в пользовательском кольце, выполняются в собственных адресных пространствах даже в отсутствие MMU (Memory Management Unit – аппаратный диспетчер памяти – отвечает за трансляцию виртуальных адресов в физические, защиту памяти, управление кэш-памятью, арбитраж на шине данных и т.п. Реализован не во всех процессорах. – *Прим. пер.*), а при наличии MMU эти адресные пространства являются защищёнными, что увеличивает надёжность.

Кроме того, модель RTP при сохранении всех существующих преимуществ VxWorks, таких как масштабируемость, производительность и предсказуемость, а также доступности всех необходимых подмножеств API (Application Programming Interface – интерфейс прикладного программирования. – *Прим. пер.*) ядра ОС обеспечивает бóльшую совместимость со стандартами POSIX и позволяет реализовать в VxWorks проверенную парадигму программирования – *процесс*. (В современной трактовке многозадачности роль задачи в ОС, то есть исполняемой единицы, подлежащей планированию, выполняет поток – thread; процесс – process при этом является сущностью пассивной, то есть представляет собой контейнер потоков с общим адресным пространством. Модель RTP полностью соответствует этой трактовке. – *Прим. пер.*)

Сравнение Linux и VxWorks

Wind River Real-Time Core реализует поддержку реального времени в Linux посредством выделения необходимых функций в независимое микроядро, имеющее свой собственный планировщик. Приложения, компонуемые с этим микроядром, работают в режиме гарантированного реального времени, и им обеспечиваются предсказуемые значения времени реакции на прерывание и переключения контекста; все остальные сервисы обеспечиваются традиционными задачами Linux. Таким образом, в распоряжении приложения оказывается больше функциональных возможностей, хотя далеко не все они будут являться сервисами реального времени.

В VxWorks, с другой стороны, в режиме реального времени работает вся система; функциональная расширяемость на прикладном уровне достигается за счёт применения процессной модели программирования с использованием вызовов POSIX API.

Резюме

Постоянно растущая сложность требований к встраиваемому ПО и многообразие доступного на рынке инструментария способны значительно усложнить жизнь современного разработчика. Решения компании Wind River позволяют выбрать соответствующий инструмент разработки приложений реального времени в зависимости от требований конкретной задачи. ●

Авторизованный перевод
Николая Горбунова,
сотрудника фирмы ПРОСОФТ
Телефон: (495) 234-0636
E-mail: info@prosoft.ru