Разработка приложений на базе операционной системы реального времени RTEMS

Николай Баландин, Александр Крапивный (Москва)

Статья посвящена операционной системе реального времени RTEMS, настройке инструментов для разработки приложений на базе RTEMS и процессу сборки системы.

Введение

Операционная система реального времени RTEMS (Real-Time Executive for Multiprocessor Systems) является некоммерческой системой, которая поставляется по модифицированной лицензии GPL, её разработка ведётся с 1988 г. американской компанией OAR Corporation (On-line Applications Research Corporation). Изначально разработка ОС велась по заказу Министерства обороны США, поэтому буква «М» в сокращении RTEMS раньше обозначала Missile и даже Military для исходных текстов, написанных на Ada. На сегодняшний день операционная система RTEMS предоставляется в виде исходных текстов, которые можно скачать с ftp://ftp.rtems.com. На начало 2010 г. последней версией RTEMS является 4.10. В данной статье будет рассмотрена последняя стабильная версия 4.9 (осень 2009 г.).

RTEMS и другие ОСРВ

Основными конкурентами RTEMS в секторе операционных систем жёсткого реального времени являются QNX, eCos и VxWorks. ОСРВ QNX поддерживает разные типы архитектур (в том числе x86), но поставляется по проприетарной лицензии. Начиная с конца 2007 г. компания QNX Software Systems приступила к постепенному открытию исходного кода QNX Neutrino под лицензией смешанного типа, которая позволяет свободным разработчикам получить доступ к исходным текстам. Но для коммерческого использования необходимо приобретать соответствующую лицензию. Компания QSS запрещает проводить сертифицирование производных от исходных текстов QNX программных продуктов без письменного разрешения компании QSS. ОСРВ VxWorks является одной из самых известных операционных систем жёсткого реального времени, но поставляется по проприетарной лицензии и имеет закрытый исходный код. ОСРВ eCos поставляется как по свободной, так и по проприетарной лицензии, и хотя утверждается, что свободная версия является стабильной и полностью протестированной, в ней не реализованы или не обновляются некоторые дополнительные возможности, например, стек сетевых протоколов TCP/IP не соответствует сегодняшним стандартам.

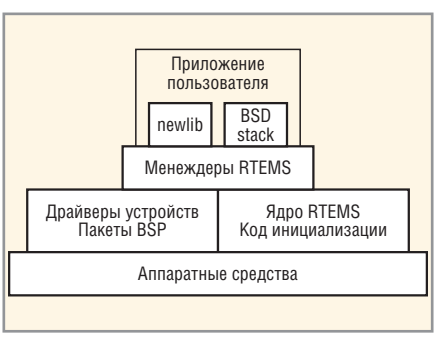


Рис. 1. Структура приложения на базе RTEMS

Концепция RTEMS

RTEMS использует принцип SUSP (single user, single process: один пользователь, один процесс, много потоков – задач, стандарт POSIX 1003.1b). Большая часть исходного кода системы является платформонезависимой, благодаря этому RTEMS поддерживает разные типы архитектур, и приложения, написанные под один тип процессоров, могут быть перенесены на другие архитектуры ЭВМ. На настоящий момент RTEMS поддерживает следующие архитектуры:

- SPARC;
- Intel 386 и выше;
- Motorola 68k;
- ARM;
- Mips
- Blackfin;
- PowerPC.

RTEMS имеет следующие возможности:

- многозадачность;
- поддержку многопроцессорных систем;
- поддержку стандартного стека сетевых протоколов TCP/IP, перенесённого с FREE BSD;
- поддержку файловых систем FAT, NFS, IMFS, TFTP;
- динамическое выделение памяти;
- большое разнообразие реализуемых механизмов синхронизации, взаимодействия задач и управления приоритетами;
- возможности оптимизации работы и размера системы при правильной настройке параметров конфигурации системы;
- наличие командного интерпретатора;
- поддержку POSIX 1003.1b, ITRON 3.0.

Структура приложения на базе RTEMS представлена на рис. 1. Помимо вышеперечисленных преимуществ, ОСРВ RTEMS имеет три основных недостатка:

- RTEMS является «глубоко встраиваемой» системой (Deeply Embedded), предполагается, что алгоритм или настройки не будут меняться, поэтому отсутствуют загружаемые модули;
- RTEMS не имеет поддержки журналируемых файловых систем (Ext3, ReiserF);
- RTEMS не располагает стандартными резидентными средствами отладки.

RTEMS может рассматриваться как набор компонентов, предоставляющих набор сервисных функций для работы приложения реального времени. Прикладной интерфейс RTEMS сформирован распределением сервисных функций по логически подобранным наборам, называемым менеджерами. Базовые функции, используемые несколькими менеджерами, такие как планирование, диспетчеризация и управление объектами, реализованы в ядре (core). Рисунок 2 показывает организацию ядра и менеджеров RTEMS.

В RTEMS 4.9.0 реализованы следующие менеджеры:

- менеджер задач (task manager) – выполняет функции создания и удаления задач и управления ими;
- менеджер прерываний (interrupt manager) – даёт быстрый ответ на внешние прерывания;
- менеджер времени (clock manager) – поддерживает дату, время и другие возможности, связанные с отчётом времени;
- менеджер таймеров (timer manager) – управляет таймерами;
- менеджер семафоров (semaphore manager) – управляет семафорами для синхронизации;
- менеджер сообщений (message manager) – обеспечивает связь и синхронизацию через очереди сообщений;
- менеджер событий (event manager) – обеспечивает связь и синхронизацию через события;
- менеджер сигналов (signal manager) – обеспечивает асинхронную связь через сигналы;
- менеджер разделов (partition manage) – обеспечивает динамическое выделение памяти блоками фиксированного размера;
- менеджер регионов (region manager) – обеспечивает выделение памяти блоками переменного размера;
- менеджер двухпортовой памяти (dual ported memory manager) – механизм преобразования адресов между внутренними и внешними областями двухпортовой памяти;
- менеджер ввода-вывода (I/O manager) – обеспечивает доступ к драйверам устройств;
- менеджер обработки неисправимых ошибок (fatal error manager);
- менеджер монотонной частоты (rate monotonic mananger) – реализует периодически выполняемые задачи, собирает статистику о выполнении;
- менеджер расширений пользователя (user extensions manager) – дополняет ядра функциями пользователя, которые будут вызываться при критических системных событиях;
- менеджер многопроцессорности (multiprocessing manager);
- менеджер инициализации (initialization manager) – инициализирует и выключает RTEMS.

В приложениях реального времени важную роль играет возможность взаимодействия и синхронизации задач. RTEMS поддерживает следующие виды синхронизации и взаимодействия:

- обмен данными между задачами;
- обмен данными между задачами и процедурами обслуживания прерываний (ISR);
- синхронизация между задачами;
- синхронизация между задачами и процедурами обслуживания прерываний.

Рассмотрим работу RTEMS на процессорах семейства Intel x86. RTEMS использует защищённый режим, плоскую модель памяти и не использует страницы. Код пользователя, как и ядра, работает на высшем уровне привилегированности Ring 0, поэтому в программе пользователя возможно выполнение привилегированных инструкций процессора. По умолчанию RTEMS обрабатывает ошибки типа fatal error следующим образом: вполняется инструкция CLI (запрещаются прерывания), в регистр eax помещается код ошибки и выполняется инструкция HLT (процессор останавливается, так как прерывание не наступит).

RTEMS использует следующие сегменты в памяти:

- один сегмент кода, который содержит весь код приложений и ядра;
- один сегмент данных, который содержит все данные приложений и данные ядра.

Для быстрого переключения задач RTEMS использует один процесс и много потоков, вследствие этого возникает опасность повредить данные, или код других потоков, или ядра при неправильной работе с указателями в одном из потоков пользователя.

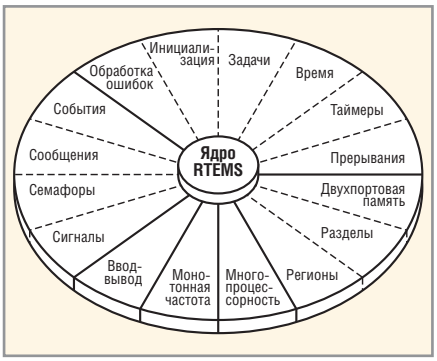


Рис. 2. Организация ядра RTEMS

Разработка

Разработка RTEMS-приложений происходит на двух машинах: основной или инструментальной (host platform) и целевой (target platform). На основной машине должна быть установлена ОС Linux, Windows, BSD или Solaris, а также средства разработки (компилятор и утилиты для сборки) и средства для кросс-компиляции (GNU tools).

Сразу стоит отметить, что разработка на Windows требует установки пакета Cygwin поддержки выполнения программ для Linux в среде Windows, при этом и так не быстрая сборка будет проходить примерно в пять раз медленнее, чем в Linux, и возможно появление непредсказуемых ошибок (производитель OAR Corp. рекомендует использовать Unix-системы). Для разработки на Linux нужно иметь достаточно много свободного места на жёстком диске: распакованные исходные тексты занимают около 80 Мб (примерно 8000 файлов), распако-

ванные инструменты для сборки – до 350 Мб, место для установки – до 200 Мб, временный каталог для сборки – до 750 Мб. После установки инструментов для сборки RTEMS должна быть собрана под определённую архитектуру процессора и должны быть установлены пакеты поддержки платы BSP (Board Support Package).

Базовыми для сборки RTEMS являются компилятор gcc (для RTEMS 4.9 версия не ниже 4.3.2) и утилита GNU make, на них устанавливаются специальные дополнения для RTEMS. Уточнить версию компилятора можно командой gcc –version. Если версия компилятора удовлетворяет требованиям, можно переходить к установке пакетов для RTEMS. Устанавливать их можно двумя способами: собирать из исходников с наложением патчей или ставить RPM-пакеты (подходит для OpenSuse, RedHat, Fedora, CentOS). Также одной из задач сообщества разработчиков RTEMS является создание пакетов для Debian.

Сборка инструментов из исходных текстов

В общем случае для сборки из исходников нужно скачать соответствующие версии архивов с ftp://ftp.rtems.com, а затем распаковать их. Для сборки необходимо поставить базовые пакеты GCC, NEWLIB и BINUTILS, затем их дополнения для RTEMS. Порядок выполнения сборки и установки:

- создаём папку tools и сохраняем в ней архивы с исходными текстами пакетов gcc, binutils, newlib, autoconf, automake;
- распаковываем исходные тексты для tar.gz и tar.bz2:

```
$ cd tools
$ tar -xzf toolname.tar.gz
$ tar -jxf toolname.tar.bz2
```

- добавляем патчи:

```
$ cd gcc-4.3.2
$ cat ../gcc-4.3.2-rtems-
20090909.diff | patch -p1
```

- те же самые действия производим и для binutils, newlib, autoconf и automake;
- начинаем последовательно сборку и установку сначала binutils, потом gcc и newlib:

```
$ mkdir build-binutils
$ cd build-binutils
$ ./binutils/configure --
target=i386-rtems-4.9 --
prefix=/opt/rtems-4.9
$ make all
$ make info
$ sudo make install
```

Для установки gcc и newlib необходимы работающие binutils, поэтому нужно указать путь к уже установленным утилитам:

```
$ export PATH=$PATH:/opt/rtems-
4.9/bin
```

Если путь /opt/rtems-4.9 не будет зафиксирован в переменной окружения PATH, то при сборке gcc и newlib возникнут ошибки. gcc и newlib собираются по такому же принципу, как и binutils.

Установка RPM-пакетов

Существенно более простым способом установки инструментов для сборки RTEMS является установка готовых RPM-пакетов. На FTP RTEMS имеются последние версии готовых пакетов для RedHat, OpenSuse, Fedora и CentOS. Порядок установки на OpenSuse 11.1 (RPM-пакеты взяты с ftp://ftp.rtems.com и помещены в папку RPMS):

```
$ cd RPMS;
$ sudo rpm -U rtems-4.9-binutils-
common-2.19-3.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-i386-
rtems4.9-binutils-2.19-
3.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-gcc-com-
mon-4.3.2-23.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-newlib-
common-1.16.0-
23.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-i386-
rtems4.9-newlib-1.16.0-
22.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-i386-
rtems4.9-gcc-4.3.2-22.su-
se11.1.i586.rpm
$ sudo rpm -U rpm -U rtems-4.9-
i386-rtems4.9-gcc-c++-4.3.2-
22.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-autoconf-
2.62-4.suse11.1.noarch.rpm
$ sudo rpm -U rtems-4.9-automake-
1.10.1-4.suse11.1.noarch.rpm
```

Пакеты нужно устанавливать в строго определённом порядке, поскольку между ними имеются зависимости. Проверить правильность установки можно, посмотрев содержимое папки /opt/rtems-4.9/ (компилятор будет находиться в /opt/rtems-4.9/bin/). Также, если необходим кросс-отладчик GDB, устанавливаем пакеты gdb-common и gdb-i386-rtems-4.9 (они требуют уже установленного отладчика GDB):

```
$ sudo rpm -U rtems-4.9-gdb-com-
mon-6.8-11.suse11.1.i586.rpm
$ sudo rpm -U rtems-4.9-i386-
rtems4.9-gdb-6.8-
11.suse11.1.i586.rpm
```

Установка исходных текстов системы

Исходные тексты последней версии системы доступны на ftp://ftp.rtems.com. Помещаем архивы rtems-4.9.0.tar.bz2 и examples-4.9.0.tar.bz2 (примеры приложений) в домашний каталог и распаковываем их:

```
$ cd ~
$ tar -jxf rtems-4.9.0.tar.bz2
$ tar -jxf examples-4.9.0.tar.bz2
```

Исходные тексты разделены на следующие папки:

- autoconf/ – содержит скрипты autoconf.ac для генерации configure;
- automake/ – содержит скрипты Makefile.am для генерации Makefile;
- c/ – часть исходных текстов, которые являются зависимыми от модели процессора и BSP;
- cpukit/ – платформонезависимая часть кода (TCP/IP, FTP, HTTP и т.д.);
- doc/ – текстовые файлы для генерации документации в форматах info, HTML, Pdf;
- make/ – файлы для поддержки Makefile-ов пользователя;
- testsuites/ – примеры программ работы с различными API и библиотеками;
- tools/ – специальные утилиты.

Стоит также отметить, что программисты, для которых привычным является графический интерфейс, могут воспользоваться средой разработки KDevelop или Eclipse в ОС Linux.

Сборка системы

Перед компиляцией необходимо исправить строку 161 файла /opt/rtems-4.9/i386-rtems4.9/include/sys/unistd.h, в ней заменяем int _EXFUN(ttyname_r, (int, char *, size_t)) на int _EXFUN(ttyname_r, (int, char *, int)). Это исправление не является исправлением ошибки

в системе, оно необходимо для компилятора gcc-4.3.2.

Экспортируем путь к компилятору:

```
$ export PATH=$PATH:/opt/rtems-4.9/bin
$ cd ~
$ mkdir build
$ cd build
$ ../rtems-4.9.0/configure --target=i386-rtems4.9 \
--enable-rtemsbsp=pc386 \
--prefix=/opt/rtems_1 \
--enable-posix \
--enable-networking \
--disable-itron \
--disable-cxx
$ make all
$ sudo make install
```

Эта команда соберёт RTEMS для процессора Intel 386 или выше с поддержкой системного интерфейса POSIX, стека сетевых протоколов FreeBSD и без поддержки системного интерфейса Itron и языка C++. Опция –-prefix отвечает за путь, по которому будет произведена установка RTEMS.

Установка производится командой make install (с правами суперпользователя).

Сборка приложений

Вместе с исходными текстами RTEMS поставляются простые примеры программ и Makefile-ов, для того чтобы разработчик смог быстро начать писать собственные RTEMS-приложения. В приложениях можно указывать такие параметры конфигурации системы, как ограничение на максимальное количество задач, таймеров, семафоров, использование определённых драйверов и т.д.

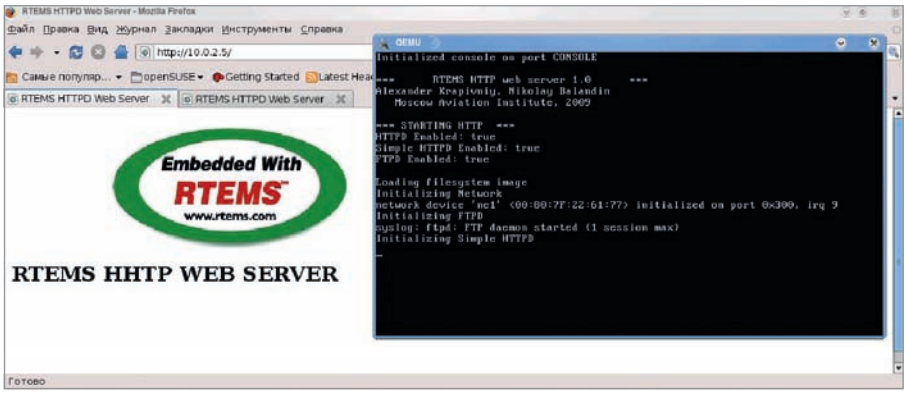


Рис. 3. Web-сервер на базе RTEMS

Перед сборкой помимо пути к компилятору i386-rtems-gcc-4.3.2 нужно экспортировать путь к установленной RTEMS, на базе которой мы хотим собрать наше приложение. Если на одной машине мы разрабатываем, например, приложения для разных процессоров или разных BSP, в каталоге /opt у нас будет установлено несколько разных версий RTEMS, и мы должны указать путь именно к той, которая нам нужна:

```
$ export RTEMS_MAKEFILE_PATH=/opt/rtems_1/i386-rtems4.9/pc386
$ make
```

В общем случае этот путь выглядит так {install_point}/{cpu_model}/{bsp}/.

В результате успешной компиляции в папке с программой file.c появляется папка o-optimize, а в ней file.exe – это и есть образ нашего RTEMS-приложения.

Запустить приложение можно как в эмуляторе Qemu, предварительно настроив его с загрузчиком GRUB, так и на целевой машине, загрузив образ file.exe при помощи GRUB с жёсткого диска, дискеты или CD-ROM. На рис. 3 показан web-сервер на базе RTEMS, запущенный в эмуляторе Qemu, и браузер Firefox на локальной машине, принимающий html-страницу с web-сервера RTEMS через виртуальную сеть.

Заключение

На сегодняшний день ОСРВ RTEMS является одной из немногих доработанных и документированных операционных систем реального времени, имеющих открытый исходный код и довольно небольшие требования к объёму памяти и быстродействию процессора, среди подобного класса систем и имеет положительный опыт применения в военных космических системах. Также стоит отметить, что в завершающей стадии находится работа по переносу исходного текста RTEMS для работы на микроконтроллерах AVR32 фирмы Atmel.

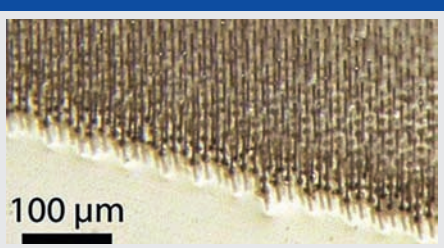
Литература

1. Getting Started with RTEMS.
2. RTEMS Applications C User's Guide.
3. RTEMS POSIX API User's Guide.
4. RTEMS Development Environment Guide.
5. RTEMS CPU Supplement.

Новости мира News of the World Новости мира

Разработаны недорогие солнечные элементы из пластика

Исследователи Калифорнийского технологического института (California Institute of Technology, Caltech) разработали гибкие солнечные элементы, поглощающие до 96% падающего света. Столь высокий результат достигнут за счёт особой конструкции элементов: на пластиковой подложке расположены вертикально-ориентированные кремниевые штырьки. Разработчики отмечают очень низкую стоимость используемых материалов, поскольку элемент на 98% состоит из пластика, а доля дорогостоящих фотовольтных материалов в 50 раз меньше, чем в традиционных батареях.



Экспериментальные образцы предлагаемого решения демонстрируют до 90% КПД, тогда как этот показатель для лучших образцов кремниевых элементов составляет всего 25%. Руководитель группы исследователей Гарри Этуотер (Harry Atwater) говорит, что разреженный массив светочувствительных волосков не только повышает площадь поглощающей поверхности, но и способствует концентрации светового потока. Кремниевые стержни или волоски толщиной всего 1 мкм и длиной от 30 до 100 мкм покрыты прозрачным пластиком. Часть светового потока, не поглощённая стержнем, отражается от подложки и попадает на другие стержни, увеличивая тем самым коэффициент поглощения. В настоящее время исследователи работают над увеличением размера нового элемента, чтобы изготавливать целые рулоны таких солнечных батарей.

Caltech