Использование «свободных» программных средств для разработки встраиваемых систем на основе 32-разрядных RISC-процессоров

Сергей Емец

Введение

Из-за высокой стоимости средств разработки существует интерес к так называемому «свободному» программному обеспечению. Термин «свободный» требует пояснения: на английском языке данные продукты называются «free software», что может быть переведено на русский либо как «бесплатный», либо как «свободный». В данном контексте смысл имеет перевод «бесплатный», но общеупотребительным является перевод «свободный». По моим данным, в России «свободный» софт не имеет такой популярности среди разработчиков встраиваемых систем, как в странах с более жесткой лицензионной деятельностью. Причиной этого является то, что в России существует достаточное количество бесплатного «пиратского» программного обеспечения, которое более популярно благодаря рекламной деятельности компаний, его производящих, литературе и навыкам работы, а также привычкам, сформировавшимся у разработчиков. Но наблюдающаяся тенденция усиления борьбы с нелегальными программами или потребность продавать продукт на внешнем рынке могут потребовать от разработчика отказа от использования «пиратских» программ. Тогда встает вопрос приобретения лицензии или использования другого программного продукта. Данная статья дает обзор некоторых свободных продуктов, которые успешно конкурируют с коммерческими программами, предназначенными для разработки программного обеспечения встраиваемых систем.

Лицензия GNU

Существует множество методов приобретения бесплатного программного обеспечения. Это могут быть версии коммерческих программ с ограниченной функциональностью или какие-либо программы, предназначенные для некоммерческого использования. В данной статье рассматривается софт, защищенный лицензией GNU (**www.gnu.org/copyleft/gpl.html**). Дополнительную информацию на русском языке можно получить на **www.gnu.org/copyleft/copyleft.ru.html**.

В общих чертах *copyleft* (сравн. с copyright) запрещает присваивать и скрывать изменения, внесенные в программу, от последующих пользователей, стимулируя таким образом развитие и улучшение программ, защищенных данной лицензией. Но в то же время продукт, полученный с помощью программы, является собственностью того, кто его получил, и может быть использован для получения прибыли. Поясню на примере компилятора GCC: если будет изменен сам компилятор так, что он будет генерировать код для нового процессора, то лицензия GNU запрещает продавать этот компилятор кому-либо; но программы, скомпилированные GCC, можно продавать, и их исходные тексты могут не предоставляться пользователям.

Преимущества, которые обеспечиваются доступом к исходным кодам

Из программ, применяемых при разработке встраиваемых систем, в данной статье рассматриваются средства программирования и отладки: компилятор (вернее, крос-компилятор), операционная система реального времени, отладчик. Тем, кто интересуется программами разработки и симуляции принципиальных схем, подготовки к производству печатных плат, языками HDL и т. п., я рекомендую посетить следующую страничку: **http://lem.ch.unito.it/linux/Z/1/index.shtml**.

Итак, об ОС, которые распространяются по лицензии GNU. Наибольшее распространение получили две операционные системы реального времени (RTOS) — RTEMS (**www.OARcorp.com/RTEMS/rtems.html**) и eCos (**www.sourceware.cygnus.com или www.redhat.com**). RTEMS произошла из проекта системы управления крылатых ракет и присутствует на рынке длительное время, что дает много плюсов, среди которых наличие готовых приложений, оттестированных для различных архитектур, например стек TCP/IP, большое количество разработчиков, использующих данную систему, и, соответственно, возможность получить в конференциях ответы на вопросы, возникающие при ее использовании.

Операционная система eCos была разработана компанией Cygnus, хорошо известной в мире свободного программного обеспечения. К программным продуктам, которые были разработаны или поддерживались данной компанией, мы будем возвращаться и далее. Сейчас Cygnus является частью Redhat — компании, которая поддерживает наиболее популярный дистрибутив Linux. Плюсами eCos являются: продуманность структуры ОС, позволяющая легко переносить (портировать) систему на различные платформы (к данному моменту по числу поддерживаемых архитектур eCos превосходит RTEMS, и это число продолжает расти — **http://sources.redhat.com/ecos/hardware.html**); наличие группы поддержки пользователей, организованной по принципу конференции — на задаваемые вопросы ответ дается в течение пары дней (во многих коммерческих проектах на получение ответа зарегистрированным пользователям требуется больше времени). Архивы конференции можно найти на странице **http://sources.redhat.com/ml/ecos-discuss**. Обе системы поставляются в виде исходных кодов, и пользователь имеет возможность вносить изменения и переносить систему на любой процессор и плату.

Еще раз подчеркну важный факт, принципиально отличающий данные операционные системы от коммерческих аналогов, — пользователь имеет полный доступ к исходникам. Это означает, что в проекте не будет темных мест в виде каких-либо объектных файлов или скомпилированных библиотек. То есть можно проходить отладчиком (и, если потребуется, вносить изменения) не только код приложения, но и код самой ОС. Добавим, что наличие исходников и специальных функций компилятора и линковщика позволяют создавать приложение с минимальным объемом требуемой памяти, исключив функции ОС, не используемые в данном приложении (обычно во встраиваемой системе получается один образ памяти, объединяющий как приложения, так и ОС). Доступ к исходным кодам приобретает исключительную ценность при разработке процессорного ядра на ПЛИС или заказной микросхемы с процессорным ядром. В этом случае «железо» бывает не всегда совместимо с промышленным стандартом, и поддержку дополнительных свойств и ресурсов (и наоборот, замену каких-либо не существующих в «железе» ресурсов софтом) возможно осуществить в ОС.

GCC, BINUTILS, GDB

Для сборки как самой операционной системы, так и приложений требуется компилятор, линковщик, преобразователь форматов для работы с образом программы (например, преобразование его из бинарного формата в Motorola S-record). В подавляющем большинстве случаев при разработке встраиваемых систем используется не компилятор, а кросс-компилятор — то есть компилятор, который работает на одной архитектуре, а код генерирует для другой.

По-видимому, для компилятора GCC не существует альтернативы: — по количеству поддерживаемых процессоров и работе оптимизатора он превосходит любой коммерческий проект. Компилятор можно разделить на три части: 1) так называемый *front-end* — программа разборки синтаксиса и генерации промежуточного кода, 2) *оптимизатор* — программа оптимизации промежуточного кода, 3) *back-end* — программа трансляции промежуточного кода в команды ассемблера. Такая структура позволяет развиваться в двух направлениях — поддержка многих языков (Фортран, Ада и т. д) и поддержка многих платформ. В первом случае вносится изменение во front-end, во втором — в back-end. Вместе с языком поставляется библиотека функций *libc*, которая является, если так можно сказать, стандартнейшей из библиотек. Для программирования на C++ можно взять стандартную C++ библиотеку *libstdc++*, в которой присутствует потоковый ввод/вывод и контейнерные классы.

Ассемблер, линкер и утилиты для работы с исполняемым кодом входят в пакет *binutils*. Он может также использоваться как для генерации «родного» (native) кода, то есть кода, исполняемого на той же архитектуре, на какой работают утилиты, так и кросс-кода, используемого при разработке встраиваемых систем.

Для отладки программ применяется многоплатформенный отладчик GDB, который позволяет отлаживать как родной, так и кросс-код. Для отладки кросс-кода может использоваться как режим симуляции, в котором команды отлаживаемого процессора исполняет симулятор, так и режим удаленной отладки (remote), в котором к отлаживаемому процессору подключается терминал. Для связи могут использоваться как стандартные интерфейсы (последовательный порт или Ethernet), так и специальные ICE-интерфейсы. Наличие исходников позволяет добавить любой канал связи. Следует отметить, что пользовательский интерфейс одинаков во всех случаях (естественно, различия в архитектуре процессоров отображаются, но набор команд и исполняемые функции одинаковы).

GDB предоставляет пользователю интерфейс командной строки, достаточно удобный и эффективный, но в то же время существуют графические интерфейсы к *gdb — xxgdb, ddd*. Также следует отметить пакет Insight от Cygnus, который включает в себя отладчик и интегрированный графический интерфейс.

Выбор «хоста»

По поводу выбора аппаратной платформы вопросов обычно не возникает. Распространенность и относительно низкая стоимость предопределяют использование IBM-совместимого компьютера. Возникает лишь вопрос выбора рабочей операционной системы. Данный вопрос, наверное, определяется индивидуально для каждого пользователя, поскольку выбор ОС определяется тем, под какую систему написаны приложения, используемые в работе. То есть если рабочий инструмент — Microsoft Word, то операционной системой должна быть, безусловно, Microsoft Windows. Со средствами, описанными выше, ситуация несколько иная. Так как они поставляются в виде исходников, то могут быть собраны для любой ОС, имеющей C-компилятор и поддерживающей требуемый набор системных функций (например, доступ к файлам). То есть свободные средства работают не только под Linux, но и под Windows. Поэтому при выборе ОС «хоста» приоритетными являются привычка или удобство использования, тем более что существуют эмуляторы Windows под Linux (wine), Linux под Windows (cygwin) и система виртуальных машин WMWare, которая позволяет запустить что угодно из-под чего угодно (последняя не является свободной программой). Мне кажется, что более удобно использовать Linux, так как среда разработки является естественной. В этом случае для разработки программ для ПК используются те же (GCC, GDB) библиотеки функций, что и для программирования и отладки встраиваемых приложений. Для Linux существуют как оконные интерфейсы, которые непосвященный пользователь не отличит от интерфейса Microsoft Windows, так и непохожие на него. Но при этом есть удобные интерпретаторы командной строки и набор стандартных утилит, которые являются очень мощным средством работы с файлами и взаимодействия с ОС.

Пользователям Windows следует обратить внимание на пакет Cygwin — продукт Cygnus, который предоставляет программам набор системных вызовов, требующихся приложениям Unix. Далее предполагается, что у пользователей Windows есть такой продукт, и различий между Windows и Linux по возможности не делается.

Установка и сборка eCos для ARM-совместимого процессора

В качестве примера рассмотрим использование eCos для программирования ARM-процессора (**www.arm.com**). В настоящее время большая часть встраиваемых приложений использует процессоры с ядром ARM. Статистику можно найти на сайте ведущего отраслевого журнала EETimes (**www.eet.com** или **www.eetimes.com**). Для работы использовалась демонстрационная плата AEB-1 (**www.arm.com/Documentation/Overviews/AEB/index.html**) с процессором Sharp LH77790. Он не имеет режима вычислений THUMB (16-бит) и использует ядро ARM7DI. Для увеличения плотности кода в разработках я бы рекомендовал использовать более новые модификации ядра ARM7TDMI, ARM9TDMI. Такие ядра встроены в чипы AT91 популярного в России производителя Atmel.

1. Установка для Linux

Процедура установки eCos для платформы Linux достаточно проста и не отличается от установки любого другого продукта, поставляемого по лицензии GNU (поскольку Cygnus является подразделением Redhat, присутствует также пакет для RPM-инсталлятора). Я обычно пользуюсь пакетами, сжатыми

bzip2, так как из-за меньшего размера получается выигрыш в скорости загрузки и стоимости соединения. Для упрощения процедуры установки одновременно с исходниками операционной системы там же можно скачать «development tools», в которые входят компилятор, отладчик и утилиты, описанные выше. Для ARM-платформы рекомендуется скачать «eCos development tools», даже если уже имеются исходники текущей версии GCC и утилит. Основная ветвь разработки GCC поддерживает COFF-формат объектных файлов для ARM, в то время как в eCos используется более новый ELF.

Для установки и сборки средств требуются следующие утилиты (при инсталляции Linux они обычно устанавливаются автоматически): GCC компилятор для «родного кода», набор стандартных утилит (*patch, make, yacc, sed, automake, lex* и т. п), архиватор (*tar*) и программа сжатия (*gzip* или *bzip2*). Процедура установки очень подробно описана на **http://sources.redhat.com/ecos/tools/linux-arm-elf.html**, я только прокомментирую командные строки и дам некоторые рекомендации (предполагая, что вы пользуетесь командным интерпретатором *bash*):

1) инсталляцию лучше проводить под пользовательским логином — это является более «культурной» практикой (хотя в рекомендациях, по-видимому, предполагается полный доступ):
   mkdir -p /src/binutils /src /gcc /src/gdb;
2) рекомендуется использовать отдельную директорию для каждого продукта (если у текущего логина нет прав записи в директорию */src* или она отсутствует, зайдите в систему root-ом и создайте эту директорию с возможностью записи в нее *mkdir /src; chmod 777 /src*) или создайте дерево в home-директории (*mkdir -p ~/src/binutils ~/src/gcc ~/src/gdb*) и в дальнейшем обращайтесь к ним не */src…*, а *~/src;*
3) скопируйте архивы в соответствующие директории;
4) для каждого архива выполните команду (находясь в соответствующей директории):
   bunzip2 < _ИМЯ_ПРОДУКТА_.tar.bz2 | tar xvf
   или
   gunzip < _ИМЯ_ПРОДУКТА_.tar.gz | tar xvf
   в зависимости от того, какие архивы были скопированы;
5) внесите изменения в дистрибутив GCC (смотри выше про COFF и ELF форматы):
   cd /src/gcc/gcc-2.95.2 (или *~/src…*)
   patch -p0 < ecos-gcc-2952.pat;
6) стандартная процедура инсталляции состоит из двух шагов: *configure* и *make*.
   Вначале следует собрать *binutils*, затем все остальное:
   mkdir -p /tmp/build/binutils
   cd /tmp/build/binutils
   /src/binutils/binutils-2.10/configure
   *--target=arm-elf *
   *--prefix=/tools *
   *--exec-prefix=/tools/H-i686-pc-linux-gnu *
   -v 2>&1 | tee configure.out
   Для того чтобы это сработало, следует создать директорию */tools* (так же как и */src*), но более правильным считается использование директорий */local* и */local/bin* (такая конфигурация получится, если опустить ключи
   --prefix=/tools
   и
   --exec-prefix=/tools/H-i686-pc-linux-gnu)
   make -w all install 2>&1 | tee make.out;
7) ыуказать командному интерпретатору, где следует искать файлы
   PATH=/tools/H-i686-pc-linux-gnu/bin:$PATH;
   export PATH
   Для того чтобы не вводить это каждый раз при подключении, нужно добавить эту строчку в файлы *.bashrc* и *.bash_profile.*
   Если программы инсталируются в директорию */local-*, убедитесь, что переменная PATH содержит путь */local/bin* (команда *echo $PATH*);
8) сконфигурируйте и соберите остальные программы.

После инсталляции «development tools» можно приступить к сборке *eCos* (достаточно инсталлировать *binutils* и *gcc*). Для этого скопируйте пакет в требуемую директорию (в рекомендации указана директория */opt*) и выполните команду

bunzip2 < ecos-1.3.1.tar.bz2 | tar xvf - eee
gunzip < ecos-1.3.1.tar.gz | tar xvf -

После ее завершения в директории будет создана такая структура.

/prebuilt
/loaders - часть программного обеспечения, загружаемого в плату
/packages - исходники (репозиторий)
/doc
/tools
/examples
readme.txt
license.txt
assign.txt
buildid.txt

Следует лишь добавить соответствующие пути, и можно приступать к работе.

2. Установка под Windows

Сразу следует оговориться что под Windows 95/98 разработчики eCos и Cygwin не гарантируют возможность сборки средств разработки (компилятора и отладчика) из исходников. Пользователям этих систем следует пользоваться следующим методом — загрузить собранные под Windows NT или Linux (в данном случае средства компилируются с помощью кросскомпилятора для Windows) исполняемые файлы. Данные скомпилированные средства в виде *exe*-файлов можно также найти в сети, но их работоспособность может гарантировать только собравший их человек.

При написании данной статьи для сборки исполняемых файлов использовалась Windows NT 4.0.

Прежде всего следует установить пакет Cygwin, который обеспечивает работу Unix-программ в среде Windows.

Процедура может быть выполнена путем установки по сети. Для этого следует запустить процедуру инсталляции **http://sources.redhat.com/cygwin/setup.exe**. Если вы считаете такой шаг опасным (в Windows отсутвует надежный механизм защиты от различных вирусов) или не имеете доступа в Интернет, то можно приобрести упомянутую программу на CD или скачать файлы по сети и устанавливать их на локальном компьютере (отметим, что защиты от вирусов этот путь также не гарантирует). Можно загрузить исходники и попытаться собрать их (но аналогов некоторых средств под Windows не существует, и как решить эту задачу, мне неизвестно).

После запуска сетевого инсталлятора предлагается выбрать сайт, с которого будет выполняться загрузка, и директорий, куда будет проинсталлирован пакет. Далее все выполняется автоматически. После успешной инсталляции на «рабочий стол» будет положен «ярлык» *Cygwin*, кликнув который вы запустите командный интерпретатор *bash.*

Несколько рекомендаций для пользователей Windows:

1) файловая система представляет собой единое дерево, начинающееся с корня /, аппаратное устройство файловых систем скрыто от пользователя (то есть диски *a:* или *c:* отсутствуют), разделитель в полном имени файла /, а не \, имена чувствительны к регистру (*name* не эквивалентно *Name*);
2) для автозавершения в *bash* нужно нажать кнопку *Tab* (наберите *gcc* и нажмите *Tab* — в результате должно появиться *gcc.exe*). Если по первым буквам однозначно определить команду не удается, то прозвучит звуковой сигнал и после второго нажатия *Tab* будет представлен список возможных завершений *Tab* (наберите *g* и нажмите *Tab*); для повторения команд используется кнопка «стрелка вверх».
3) основные команды: *cd* — сменить каталог, *pwd* — показать текущий каталог, *ls* — список файлов, *cp* — копировать, *mv* — переместить. Для того чтобы получить инструкцию по использованию, нужно набрать: *man _command_name_* (например, *man ls*).

Запускать исполняемые файлы можно либо из командной строки MS-DOS, либо из меню *Start/Run* (Пуск/Выполнить). Однако, на мой взгляд, *bash* гораздо удобнее, и далее используется командная строка *bash.*

Для доступа к файлам, расположенным на диске *c:*, следует смонтировать файловую систему

mount -f c:/ /c,

после этого можно посмотреть файлы диска *c:* командой *ls /c.*

Далее процедура инсталляции выглядит так же, как и для Linux (но так как в Windows отсутствует система защиты, то проблем с правами доступа не возникает). Отличие состоит в том, что нужно применить патч к *insight*, и из-за того, что в DOS/Windows в отличие от Unix перевод строки кодируется двумя символами, появляются дополнительные команды *tr*:

tr -d '\r' < insight-tcl.pat | patch -p0

и

tr -d '\r' < ecos-gcc-2952.pat | patch -p0.

После завершения процедуры инсталляции отладочных средств (к этому моменту уже ин-

сталлированы компилятор, линкер, ассемблер, отладчик и утилиты) нужно загрузить инсталлятор операционной системы (я рекомендую воспользоваться зеркалами сайта (mirror)).

Процедура установки отладочных средств под Windows подробно описана в http://sources.redhat.com/ecos/install-windows.html. Следуя данной рекомендации, мне удалось собрать все, кроме графического интерфейса к gdb, но для Linux все средства собираются без проблем. Собирать средства под Windows на FAT диске не следует так как во время компиляции создается множество маленьких файлов и из-за неэффективности файловой системы FAT требуется очень много места (в данном случае требуемый объем непосредственно зависит от размера кластера). Также из-за ограничений на длину имени файла в FAT (я все-таки решил использовать FAT?, чтобы проверить сборку в наихудшем случае) возникла вышеупомянутая проблема, которую я решил, переименовав некоторые библиотеки и выполнив несколько шагов *make* вручную. ■

Продолжение следует