Для цитирования: Шалаев М. А., Аникина А. О. Проблемы автоматизированной сборки пакетов в дистрибутив класса Debian на платформе «Эльбрус» // Вопросы радиоэлектроники. 2018. № 2. С. 55–58.
УДК 004.4'2

М. А. Шалаев[1], А. О. Аникина[1]

[1] АО «МЦСТ»

ПРОБЛЕМЫ АВТОМАТИЗИРОВАННОЙ СБОРКИ ПАКЕТОВ В ДИСТРИБУТИВ КЛАССА DEBIAN НА ПЛАТФОРМЕ «ЭЛЬБРУС»

Описывается понятие дистрибутива и кросс-сборки программного обеспечения для формирования целевой системы под аппаратную платформу, отличающуюся от той, где производится компиляция. Рассматриваются основные проблемы сборки и портирования пакетов дистрибутива семейства Debian на аппаратную платформу «Эльбрус». Ставится проблема портирования ограниченным количеством разработчиков дистрибутива, развиваемого для остальных архитектур крупными группами и множеством энтузиастов. Предложены методы и средства устранения типичных ошибок и особенности реализации системы кросс-компиляции. Подробно изложены пути автоматизированного разрешения простых, сложных, а также кольцевых зависимостей. Описывается процесс автоматизации сборки, повышающий эффективность пополнения дистрибутива новыми пакетами. Реализация для системы кросс-сборки может быть использована разработчиками для пополнения пакетной базы и формирования полноценного дистрибутива на аппаратную платформу «Эльбрус».

Ключевые слова: *кросс-компиляция, сборка программных пакетов, аппаратные платформы, кольцевые зависимости, портирование дистрибутива.*

Введение

Развитие и внедрение аппаратных платформ на базе микропроцессоров семейства «Эльбрус» требует постоянного расширения, актуализации и обеспечения надежности, а также программного обеспечения (ПО), способного решать большой круг задач. Современные тенденции развития ПО, а также средств лицензирования и распространения исходных кодов заставляют обратиться к проектам с открытым исходным кодом (Open Source). Одним из ключевых решений в этой области являются дистрибутивы, созданные на базе ядра Linux. Дистрибутив операционной системы в современном понимании – это не только набор базовых программ для инициализации ОС, но также библиотеки программ, дающих возможность решать широкий круг задач. Дистрибутив формируется системой сборки, позволяющей образовать из отдельных разнородных программ единый законченный продукт [1]. Создание дистрибутива для определенной компьютерной платформы базируется на технологии кросс-компиляции, предполагающей использование одного окружения (host) при компиляции ПО для другого (target), которое может иметь иные архитектуру процессора или операционную систему [2].

На начальном этапе распространения дистрибутивы ОС могли поместиться на один компакт-диск совместно с исходным кодом; такие дистрибутивы создавались небольшими группами программистов. С течением времени размеры дистрибутивов росли, увеличивалось количество их создателей, и ситуация в корне изменилась. Например, в настоящее время дистрибутив Debian на базе ОС Linux развивают более 400 групп разработчиков и несколько сотен отдельных программистов, поддерживающих наборы пакетов в его составе. Не стоит забывать и о пользователях-энтузиастах, тестирующих обновления и использующих дистрибутив до того, как он признается стабильным. При этом средний цикл разработки релиза до официального выпуска стабильной версии составляет около двух лет, после чего продукт находится на сопровождении и получает регулярные обновления безопасности и версий ключевых пакетов.

Таким образом, естественная потребность расширить и облегчить доступ пользователей к базирующемуся на Linux программному обеспечению стала одним из основополагающих вкладов в современные компьютерные технологии, реализация которого требовала постоянно растущих ресурсов. Не сопоставляя масштабы, можно констатировать, что подобная тенденция в общем свойственна и разработке дистрибутива ОС «Эльбрус», формируемого в классе Debian.

К настоящему времени для дистрибутива ОС «Эльбрус» было портировано значительное

количество компонентов, в том числе в таких направлениях, как: графические оболочки, базы и системы хранения данных, мультимедиа, высокопроизводительные вычисления, редакторы кодов и документов, взаимодействие с периферийными устройствами, серверные и клиентские приложения.

Применительно к предыдущему состоянию эльбрусовских проектов эти результаты в определенной степени соответствовали требованиям наличного контингента пользователей. Однако в дальнейшем перед довольно немногочисленной командой программистов, связанных с разработкой и сопровождением дистрибутива, встал ряд проблем, важнейшей из которых была эффективность пополнения его пакетной базы. Это естественный фактор развития, обусловленный постоянным и быстрым ростом номенклатуры и производства передовых вычислительных средств компании АО «МЦСТ», появлением новых сфер их применения и приложений. В качестве перспективного решения был сформулирован и поставлен на реализацию представленный в данной статье принцип автоматизации процесса сборки пакетов в дистрибутив.

Особенности дистрибутивов класса Debian

Исходные тексты пакетов дистрибутива Debian включают в себя не только фиксированные версии оригинальных пакетов от их авторов, но и патчи разработчиков дистрибутива как целого. Обычно они исправляют ошибки, вносят дополнительные функции или меняют программу так, чтобы она соответствовала условиям работы в дистрибутиве. Помимо этого, с пакетами связаны так называемые rules – скрипты, которые описывают правила сборки пакетов. В дистрибутивах на базе Debian из одного пакета исходных кодов может быть сформировано несколько бинарных пакетов, устанавливаемых в систему. Скрипты сборки также перечисляют установочные бинарные пакеты.

При портировании дистрибутива Debian на платформу «Эльбрус» и адаптации его сборочных механизмов к системе кросс-сборки основные проблемы были вызваны следующими факторами:

- Разработчик Debian-пакета не предусматривает возможность его сборки компилятором, отличным от gcc: в сценариях сборки либо в Makefile оригиналов пакетов gcc указан в качестве компилятора без возможности переопределения через переменные среды.
- Разработчик пакета предусматривает использование абсолютных путей для файлов, переменных среды и флагов компиляции, которые могут использоваться при ручной или автоматизированной сборке.
- Поскольку Debian не предоставляет возможность провести полную последовательную сборку всех пакетов, необходимо решать проблему кольцевых зависимостей – это известная задача, решение которой не формализовано и не имеет полноценной автоматической реализации.
- В репозитории находятся пакеты, созданные сообществом разработчиков Debian с использованием особенностей gcc, не поддерживаемых стандартом C/C++, например nested functions.
- Набор зависимостей между пакетами не во всем соответствует ОС «Эльбрус». Например, в Debian проводится сборка пакетов с экспериментальными возможностями, которые поставляются в виде отдельных бинарных пакетов. При этом такие пакеты не востребованы функциями ОС «Эльбрус», но требуются в виде сборочных зависимостей в дальнейшей сборке дистрибутива.

Существует ряд других, более частных проблем, которые потребовалось решить при переносе дистрибутива Debian в кросс-систему сборки дистрибутива ОС «Эльбрус». Однако для возможности разработки, в которой оперативно используются материалы сообщества, фундаментальной проблемой является автоматизация включения новых пакетов [3], исключающая образование кольцевых зависимостей.

Автоматизированная система сборки дистрибутива

Разрешение кольцевых зависимостей

Дистрибутив развивается и со временем включает все больше пакетов. В настоящее время новые компоненты добавляются специалистами вручную вне зависимости от сложности портирования пакетов на платформу «Эльбрус». При этом пакетные зависимости определяются и разрешаются вручную. Для сокращения времени на пополнение дистрибутива возникла идея автоматизации этого процесса. Были проанализированы два возможных способа ее реализации:

1. Изначальное создание полного упорядоченного списка пакетов с учетом их зависимостей.
2. Итеративная сборка пакетов, в которой корректировка зависимостей определяется в процессе работы.

В первом случае необходимо составить линейный список пакетов в нужном порядке. С этой целью требуется определить все зависимости для каждого пакета из требуемого набора и построить их граф, далее с помощью алгоритма поиска в глубину определить кольцевые зависимости. В общем случае

устранить зависимости может только разработчик вручную, иначе возможны ошибки на этапе сборки. В результате устранения зависимостей формируется требуемый полный список. В этом процессе полезны существующие вспомогательные утилиты типа apt-rdepends, apt-cache, dpkg, graphviz и т.п.

Такой вариант применим при работе с небольшими группами пакетов. Когда речь идет о множестве пакетов, объединенных кольцевыми зависимостями, анализ всех вариантов разрыва колец циклов занимает неприемлемое время с учетом того, что в сборочных скриптах и управляющей информации, за редкими исключениями, нет данных о возможных разрывах циклов.

Во втором случае разрешение кольцевых зависимостей происходит в процессе сборки итеративно. На начальном этапе сборка запускается по общему списку всех необходимых пакетов. Если у пакета встречаются отсутствующие зависимости, то он фиксируется в отдельном файле, и сборка рекурсивно переходит на зависимости. В случае, когда в пакете обнаруживаются зависимости, которые уже были занесены в формируемый набор, текущему пакету добавляется опция сборки, которая отключает контроль по зависимостям. Данный пакет выступает узлом, в котором кольцо разрезается.

Рассмотрим группу пакетов pack{A-D}, связанных зависимостями:

1. $packA \rightarrow packB, packC.$
2. $packB \rightarrow packD.$
3. $packD \rightarrow packA.$

В представленной группе имеется связь, замыкающая зависимости в кольцо: packA → packB → packD → packA

При обнаружении подобной последовательности для пакета packD предпринимается попытка собрать его с уменьшенным набором зависимостей, из которого исключен packA. В случае успеха сборка продолжается автоматически. Если packD описанным образом собрать не удается, для разработчика выводится найденная цепочка, чтобы он исправил ее другим способом: разорвал цикл в другом месте или внес дополнительные изменения в первоначальный packD для того, чтобы он мог собраться с уменьшенным набором зависимостей.

Реализация сборки

Для применения рекурсивного модуля в текущей системе сборки необходимо было изменить способ установки и использовать базу пакетов, которую создает утилита apt-get. На данный момент установка пакетов в изолированное окружение для явного разделения target- и host-зависимостей производится через утилиту dpkg. Чтобы разрешать кольцевые зависимости при сборке пакета, к dpkg добавляется опция -d, позволяющая игнорировать неудовлетворенные зависимости. Это также бывает необходимо, если дистрибутив собирается с нуля и изначально нет никаких установленных пакетов. К особенностям сборки можно добавить то, что в дистрибутиве сборка пакетов производится в общем окружении, а не в индивидуальном для каждого пакета. Исходя из этого нет возможности заменить использование утилиты dpkg на apt. Поэтому было решено использовать идею, согласно которой работает утилита apt, но применять собственные наработки функций и скриптов.

Для получения новой функциональности необходимо переконфигурировать структуру системы сборки и использовать собственный репозиторий пакетной базы. С этой целью необходимо изначально адаптировать систему так, чтобы у каждого пакета было изолированное окружение, в котором не возникало бы конфликтов. Путем такой разделенной сборки получаются бинарные пакеты, не зависящие от состояния сборочного окружения, и поэтому результирующие пакеты не будут отличаться даже от варианта, когда они собираются после сборки всего дистрибутива. То есть на процесс сборки будут влиять только пакеты, явно указанные в зависимостях, а пакеты, необходимые для создания окружения, используются из уже собранной пакетной базы дистрибутива ОС «Эльбрус». Благодаря вспомогательным скриптам не возникнет ситуации, когда для сборки пакета необходимо установить ряд зависимостей, которые еще не собраны в дистрибутиве. В дальнейшем можно будет использовать уже созданные алгоритмы для автоматизации пополнения пакетной базы.

Заключение

Представленный в данной статье способ оптимизации процесса сборки пакетов в дистрибутив, который сейчас находится в стадии отладки, должен привести к сокращению времени, необходимого специалистам для пополнения дистрибутива, что будет способствовать увеличению числа добавляемых в его базу пакетов, упрощению обновлений и перехода на новые версии.

СПИСОК ЛИТЕРАТУРЫ

1. OpenSUSE Build Service. Available at: http://build.opensuse.org/ (accessed 05.11.2017)
2. Mankinen V., Rahkonen V. Cross-Compiling tutorial with Scratchbox. 2005. Available at: http://www.scratchbox.org/download/files/sbox-releases/1.0/doc/tutorial.html (accessed 05.11.2017)

3. Jackson I., Schwarz Ch., Di Carlo A., Hertzog R., Rodin J. Debian Developers Reference. 2017, 85 p. Available at: https://www.debian.org/doc/manuals/developers-reference/developers-reference.en.pdf (accessed 05.11.2017)

ИНФОРМАЦИЯ ОБ АВТОРАХ

Шалаев Михаил Андреевич, начальник отдела, АО «МЦСТ», 119334, Москва, ул. Вавилова, д. 24, тел.: 8 (495) 363-96-65, e-mail: mikhail.a.shalaev@mcst.ru.
Аникина Алена Олеговна, инженер-программист 2-й категории, АО «МЦСТ», 119334, Москва, ул. Вавилова, д. 24, тел.: 8 (495) 363-96-65, e-mail: alena.o.afanasieva@mcst.ru.

For citation: Shalaev M.A., Anikina A.O. Problems of automated assembly of packages in the Debian family distributions on the Elbrus hardware platform. Voprosy radioelektroniki, 2018, no. 2, pp. 55–58.

M. A. Shalaev, A. O. Anikina

PROBLEMS OF AUTOMATED ASSEMBLY OF PACKAGES IN THE DEBIAN FAMILY DISTRIBUTIONS ON THE ELBRUS HARDWARE PLATFORM

The concept of distribution and cross-assembly of software for the formation of a target system for a hardware platform is described. The main problems of assembling and porting packages of the Debian family distribution to the hardware platform Elbrus are considered. A limited number of developers distribution is one of the problems. Methods and means of eliminating typical errors and features of implementing a cross-compiling system are proposed. Ways of automated resolution of simple, complex and ring dependencies are detailed. The process of assembly automation which increases the effeciency of replenishment of the distribution package with new packages is described. The implementation for the cross-assembly system is described. This can be used by developers to make up the package database and form a full-scale distribution on the hardware platform Elbrus.

Keywords: cross compiling, building program packages, computer platforms, ring dependencies, porting the distribution.

REFERENCES

1. [OpenSUSE Build Service]. Available at: http://build.opensuse.org/ (accessed 05.11.2017)
2. Mankinen V., Rahkonen V. [Cross-Compiling tutorial with Scratchbox]. 2005. Available at: http://www.scratchbox.org/download/files/sbox-releases/1.0/doc/tutorial.html (accessed 05.11.2017)
3. Jackson I., Schwarz Ch., Di Carlo A., Hertzog R., Rodin J. [Debian Developers Reference]. 2017, 85 p. Available at: https://www.debian.org/doc/manuals/developers-reference/developers-reference.en.pdf (accessed 05.11.2017)

AUTHORS

Shalaev Mikhail, head of Department, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (495) 363-96-65, e-mail: mikhail.a.shalaev@mcst.ru.
Anikina Alena, 2nd category software engineer, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (495) 363-96-65, e-mail: alena.o.afanasieva@mcst.ru.