МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ

УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Инсталляция программного обеспечения в Linux

Утверждено Редакционно-издательским советом университета

в качестве методических указаний к лабораторной работе

С А М А Р А

Издательство СГАУ

2010

УДК 004.451

Составитель А.М. С у х о в

Рецензент: к.т. н., доц. Попов С.Б.

Инсталляция программного обеспечения в Linux: Методические указания к лабораторной работе/ Сост. А.М. Сухов. - Самара: Изд-во Самарского государственного аэрокосмического университета, 2010. 23 с.

В настоящих методических указаниях приведен материал, необходимый для выполнения лабораторных работ по дисциплинам «Операционная система Linux на высокопроизводительных кластерах», «Перспективные информационные технологии»

Цель лабораторной работы - научиться устанавливать новое программное обеспечение, как при помощи специальных программ инсталляторов, так и путем компилирования из исходных файлов.

Предназначено для слушателей ФПК СГАУ и студентов специальностей 010500, 010501

1. Цель лабораторной работы

- научиться устанавливать новое программное обеспечение, как при помощи специальных программ инсталляторов, так и путем компилирования из исходных файлов

- знать основные клоны Linux и различия при инсталляции, связанные с особенностями дистрибутивов

-

2. Основные сведения

Одна из первых вещей, на которые вы обращаете внимание при установке Linux -- это большое количество входящих в дистрибутив пакетов. Большинство дистрибутивов содержат операционную систему Linux, средства для инсталляции и средства администрирования. Кроме того, в них включаются средства для работы в Интернете, средства разработки, офисные пакеты, игры, а также некоторые средства, о которых вы даже не слышали. Дистрибутивы Linux, содержащие тысячи доступных пакетов, не редкость. Если вы не выбрали "установить все", будет установлено некоторое подмножество этих пакетов.

Теперь у вас могут возникнуть вопросы "Как удалить ненужные пакеты? Как установить что-то недостающее? Могу ли я использовать программное обеспечение, не входящее в мой дистрибутив?"

Пакеты RPM

В ходе установки Linux вы, вероятно, обратили внимание на информацию об устанавливаемых пакетах RPM. *RPM*, сокращение от Redhat Package Manager, созданный в Red Hat, стал стандартным средством управления программным обеспечением для Red Hat и UnitedLinux, а также для многих других дистрибутивов.

RPM -- в сущности пакет, содержащий программное обеспечение для Linux, готовое для установки и запуска на компьютере определенной архитектуры. Все программное обеспечение, изначально входящее в ваш дистрибутив, устанавливается из RPM.

Структура RPM

RPM -- это набор файлов. В него входит файл *.spec*, предоставляющий информацию о пакете, его назначении и зависимостях (какие пакеты должны быть установлены, чтобы этот пакет мог работать). Файл *.spec* также содержит манифест файлов пакета, информацию о том, куда в системе эти файлы должны быть загружены и какие они имеют изначальные права доступа. RPM также содержит преинсталляционный скрипт, написанный разработчиком пакета. Кроме того, RPM содержит скомпилированные бинарные файлы. И наконец, RPM содержит постинсталляционный скрипт.

Структура RPM

.spec	преинсталляционный скрипт	бинарный файл	бинарный файл	...	бинарный файл	постинсталляционный скрипт

При установке RPM система сначала проверяет, удовлетворяются ли имеющиеся зависимости. Если нет, процесс инсталляции прекращается, за исключением случаев, когда используются специальные опции, вынуждающие осуществить инсталляцию.

Если все прошло без проблем, запускается преинсталляционный скрипт. Это скрипт может выполнять какие угодно действия. Обычно он создает пользователей и каталоги. Однако он может генерировать различные типы динамической конфигурации, даже специально скомпилированный исходный код для запуска системы.

Что изменяют RPM

В ходе установки пакетов RPM происходит копирование файлов в систему и выполнение скриптов. Поскольку RPM запускает root, все эти действия доступны только root'у. Поэтому, прежде чем устанавливать пакет в систему, важно знать его происхождение. Так же, как и в случае с программным обеспечением для Windows, в RPM может быть включен враждебный программный код. Пакеты RPM от производителей как правило безопасны, но будьте осторожны при загрузке и установке пакетов, имеющих неизвестное происхождение.

Если преинсталляционный скрипт выполнен успешно, бинарные файлы копируются в систему в соответствии с манифестом пакета. Когда все файлы будут скопированы и права на них установлены, запустится постинсталляционный скрипт. Этот скрипт тоже может делать что угодно.

Как только процесс закончится, информация о пакете добавится в базу данных RPM, и инсталляция закончится. Этот простой метод позволяет выполнить все действия, которые могли бы быть выполнены при помощи более тщательно разработанного коммерческого инсталлятора.

База данных RPM

База данных RPM -- элемент, добавляющий пакетам RPM элегантности. Обычно эта база данных хранится в каталоге /var/lib/rpm и содержит информацию о каждом установленном в системе пакете. База данных знает о взаимных зависимостях между пакетами, и от нее поступит предупреждение, если удаление пакета может привести к повреждению других пакетов. База данных знает обо всех файлах, которые изначально установлены при инсталляции пакетов, и знает их изначальное состояние в системе. Ей также известно местоположение документации и конфигурационных файлов для каждого пакета. Может показаться, что информации очень много, и это действительно так. Но она не является раздутой и громоздкой. В системе, содержащей 1,066 пакетов, включающих в себя 203,272 файлов, файлы базы данных занимают всего 45 MB! RPM использует базу данных для проверки зависимостей между пакетами, а также при загрузке и выгрузке пакетов. Кроме того, к базе данных за информацией о пакетах могут обращаться пользователи.

Использование RPM

Программа для работы с пакетами RPM имеет соответствующее название, *rpm*. Команда *rpm* может использоваться для выполнения нескольких различных действий,

но наиболее распространенные задачи - это инсталляция, обновление, запрос, подтверждение и удаление.

 # rpm -i (install)

При установке пакета в первый раз используется опция *–i* (install). Просто укажите команде *rpm* в качестве аргумента бинарный пакет. Пакет установится в систему. Процесс инсталляции обычно занимает секунды. При установке пакета часто добавляют опцию *–v* (verbose), которая обеспечивает вывод подробной информации о ходе процесса, а также опцию *–h* (hash bar), которая по ходу выполнения процесса установки пакета выводит на консоль знаки диез (#). Вот пример инсталляции пакета:

Инсталляция пакета MyPackage

$ rpm -ivh MyPackage-1.0.0.i386.rpm

Preparing... ### [100%]

1:MyPackage ### [100%]

Теперь MyPackage установлен и готов к использованию.

Использовать *rpm* для установки и удаления пакетов может только *root*, поскольку необходим доступ файловой системе и базе данных *rpm*.

#rpm -e (erase)

Для удаления установленного пакета используется опция *-e*, 'стирающая' его. *rpm* обратится к базе данных, чтобы удалить все файлы данного пакета. Если в системе есть пакеты, имеющие зависимости от удаляемого пакета, выполнение команды *rpm* прервется. Вы можете принудительно удалить пакет, воспользовавшись опцией *nodeps*. (nodeps может также использоваться для принудительной инсталляции.) При использовании опций принудительной установки или удаления пакета будьте *крайне* осторожны. Удаление пакетов, от которых зависят другие пакеты, может иметь плачевные последствия. Ниже приведена команда для удаления ранее установленного пакета:

$ rpm -e MyPackage

Обратите внимание, что при удалении пакета не требуется указывать его полную версию. Полное наименование пакета было необходимо при его установке, поскольку мы указывали имя файла. На установленные пакеты обычно ссылаются только по их имени. Имя пакета - это все вплоть до номера версии.

#rpm -V (verify)

Опция verify очень полезна. Она сравнивает текущее состояние установленных файлов пакета с их изначальным состоянием. Для отображения различий используются специальные обозначения:

Результаты проверки файлов

S	Различается размер файлов (Size)
M	Различается состояние (Mode) (включая права доступа и тип файла)
5	Различается сумма MD5
D	Не совпадают major/minor-номера
L	Не совпадает путь readLink(2)
U	Не совпадает имя пользователя
G	Не совпадает группа
T	Различается mTime

Если применив к пакету команду *rpm –V* вы обнаружили, что изменился размер исполняемых файлов, это может быть признаком нарушения безопасности.

#rpm -U (upgrade)

Если пакет установлен, любая попытка установить пакет с тем же именем выдаст сообщение, что такой пакет уже установлен. Если вы хотите обновить пакет до более поздней версии, используйте опцию *-U*. При одновременном обновлении нескольких пакетов будет предпринята попытка установить их в таком порядке, чтобы удовлетворялись взаимные зависимости. Другими словами, первыми установятся пакеты, от которых зависят другие. Опция *–U* может использоваться и в случае, когда пакет не установлен. Многие используют эту опцию не только для обновления, но и и для установки (вместо *-i*). Ниже приведен пример использования опции upgrade для установки нескольких пакетов:

Обновление взаимозависимых пакетов

$ rpm -Uvh My.rpm*

Preparing... ### [100%]

1:bMyPackageDep ### [50%]

1:aMyPackageNew ### [100%]

В приведенном примере пакет bMyPackageDep был необходим для aMyPackageNew, поэтому *rpm* сначала установил пакет bMyPackageDep, а затем aMyPackageNew, несмотря на то, что имена файлов были расположены в обратном порядке.

#rpm -q (query)

Из базы данных *rpm* может быть получена полезная информация. Запросы может делать любой пользователь, имеющий права на чтение базы данных *rpm*. По умолчанию право на чтение имеют все пользователи. Для запроса используется опция *-q*, затем указывается имя пакета. Ответом будет версия пакета.

$ rpm -q MyPackage

$ MyPackage-1.0.0

Имя пакета должно быть указано точно. Использование метасимволов не допускается. Однако если вы не помните полное имя пакета, можете воспользоваться командой *grep*, которая поможет найти нужный пакет. Воспользуйтесь опциями *–qa* для запроса обо всех установленных в системе пакетах, а затем передайте информацию при помощи *pipe* команде *grep* с последующим указанием той части имени пакета, которую помните.

grep - средство поиска в тексте, имеющее широкие возможности. По умолчанию при поиске в файлеgrep показывает те строки файла, которые содержат искомый текст. В нашем примере мы будем искать "IBM". grep - мощный инструмент, используемый при создании скриптов и работе в консоли.

$ rpm -qa | grep IBM

IBMWSAppDev-Product-5.0-0
IBMWSSiteDevExp-Core-5.0-0
IBMWSSiteDev-Core-5.0-0
IBMWSTools-WAS-BASE-V5-5.0-0
IBMJava118-SDK-1.1.8-5.0
IBMWSWB-samples-5.0-0
IBMWSWB-5.0-0
IBMWSAppDev-Core-5.0-0
IBMWSAppDev-5.0-0
IBMWSTools-5.0-0

Помимо номера версии, *rpm –q* может выдавать и другую полезную информацию о пакете. Ниже приведены несколько примеров:

Получение информации при помощи *rpm -q*

rpm -q changelog	Показывает историю изменений при разработке пакета
rpm -qc	Показывает конфигурационные файлы для пакета
rpm -qd	Показывает файлы документации для пакета
rpm -qi	Показывает описание пакета

rpm -ql	Показывает список файлов пакета
rpm -qR	Показывает зависимости для пакета

Опция query используется и в других интересных командах, которые применяются не к пакетам, а к файлам.

$rpm -q whatprovides <filename>

Приведенная выше команда определит, какой пакет связан с данным файлом. Имя файла должно включать абсолютный путь до файла, так как именно в таком виде информация хранится в базе данных rpm.

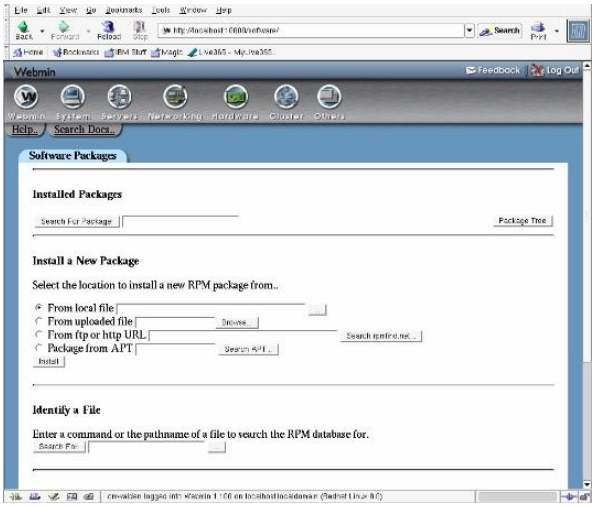
Решения для RPM с графическим интерфейсом

Работа с *rpm* в консоли не сложна, но иногда бывает удобнее использовать графический интерфейс. В Linux обычно присутствуют front-end программы, предоставляющие интерфейс для программы *rpm*. В каждый дистрибутив включен свой front-end, и они могут различаться. Информацию об используемых в вашем дистрибутиве средствах управления пакетами вы найдете в документации к дистрибутиву.

Программные пакеты Webmin

Webmin также предоставляет несложный основанный на Web-интерфейсе front-end для взаимодействия с пакетами RPM.

Рисунок 1. Интерфейс Webmin



С помощью этого средства вы можете легко установить, удалить программное обеспечение и получить информацию о нем. Кроме того, программное обеспечение может быть установлено непосредственно с сайта. Если в вашей системе установлены такие инструменты, как apt или Red Hat Network, Webmin соберет их и предоставит интерфейс доступа к ним.

Исходный код

Поскольку Linux является операционной системой с открытыми исходными кодами, он поставляется со всеми средствами разработки, необходимыми для компиляции программного обеспечения. Несмотря на то, что большинство используемых пакетов предоставляются в виде бинарных RPM, вы не ограничены только этими пакетами. При желании вы можете загрузить сырой исходный код и скомпилировать его для своей системы.

При компиляции из исходных кодов на production-системах следует проявлять осторожность, поскольку это может вызвать некоторые проблемы или прекращение поддержки коммерческого программного обеспечения, используемого в вашей системе, например, IBM DB2. Однако умение компилировать из исходных кодов позволит вам прикладывать к программам патчи и работать с пакетами, перенесенными из других окружений. Стоит вам провести успешную компиляцию из исходников, и вы сможете создать даже свой собственный RPM!

Демонстрация компиляции из исходников на примере игры Corewars

Для демонстрации того, насколько прост процесс компиляции из исходных кодов, скомпилируем моделирующую игру под названием Corewars. Вот информация о Corewars с соответствующего сайта: "Corewars -- моделирующая игра, в которой на виртуальном компьютере воины пытаются разбить друг друга. Воины могут быть созданы при помощи одного из двух ассемблер-подобных языков, Corewars или Redcode. Corewars - язык, используемый по умолчанию, он проще для изучения и понимания. Redcode предоставляет более продвинутые и развернутые инструкции, но и требует большего времени для изучения".

Первый этап компилирования из исходных кодов - загрузка пакета исходников с веб-сайта:

- http://prdownloads.sourceforge.net/corewars/corewars-0.9.13.tar.gz?download

Загрузив пакет, я распаковываю архив.

$tar -xvzf corewars-0.9.13.tar.gz

Распаковывание файлов производится в мой текущий каталог. Стандартный подход -- размещать исходные коды в каталоге с названием, соответствующем наименованию продукта. В нашем случае это каталог corewars-0.9.13.

Я перехожу в этот каталог и нахожу там исходные коды, документацию, конфигурационные скрипты и файлы README. Большинство пакетов исходников имеет файл INSTALL и файл README. Прочтите их, прежде чем приступить к компиляции. Обычно это позволяет сохранить ваши усилия, поскольку вы получите

информацию о том, как предотвратить возникновение возможных проблем, и получите инструкции по корректному проведению компиляции и инсталляции. Большинство проблем, которые возникали у меня при компиляции из исходников, были связаны с тем, что я не следовал указаниям.

Чаще всего следующим этапом идет запуск скрипта *configure*. *configure* - часть пакета Autoconf, входящего в средства разработки вашего дистрибутива Linux. Вот цитата из описания пакета Autoconf: "Autoconf -- средство GNU для конфигурирования исходных кодов и Makefile'ов. При помощи Autoconf программисты могут создавать компактные и настраиваемые пакеты, поскольку автор пакета может предусмотреть различные опции конфигурации".

Скрипт *configure* запускает в системе ряд тестов, чтобы определить оптимальный способ компиляции пакета для вашего дистрибутива и архитектуры. Затем он создает *Makefile* специально для вашей системы. При возникновении проблем компиляции скрипт *configure* выдаст соответствующее сообщение. Обычно *configure* позволяет выбрать возможности, которые вы хотите включить при компиляции или задать параметры, указывающие местоположение библиотек или необходимых файлов, чтобы сборка пакета прошла успешно. Сейчас мы запустим *configure* без дополнительных параметров:

$./configure

Несколько тестов, выполняемых в системе, наконец успешно завершились. Теперь создадим программу при помощи

$make

Если сборка прошла с ошибками, мне необходимо выявить проблемы и решить их. Это задача может быть нетривиальной и может потребовать определенных знаний о вашем окружении и программировании в целом. Если все прошло успешно, мы обычно устанавливаем программу при помощи следующей команды:

$make install

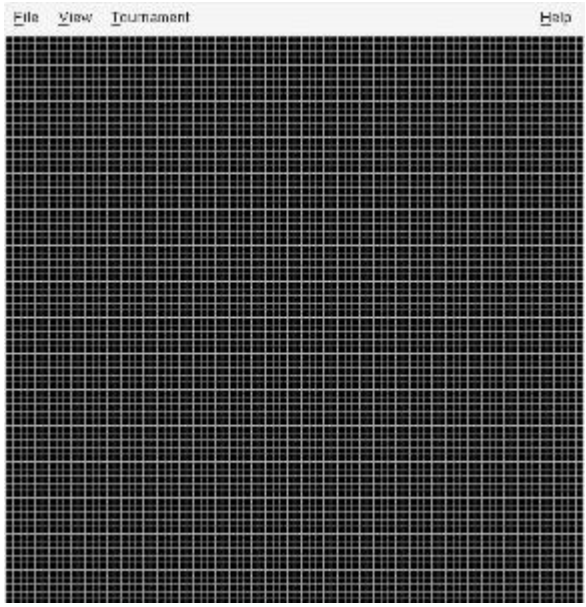
Файлы копируются в соответствующие области системы, права доступа к файлам обновляются, конфигурационные файлы копируются и документация добавляется в справочные страницы.

Теперь давайте протестируем наше изделие, запустив программу. Это графическая программа, поэтому для ее запуска нам понадобится графическая сессия. Выполненная нами команда *make install* должна поместить программу в наш путь поиска исполняемых файлов.

$corewars

В награду должен появиться графический экран.

Рисунок 2. Удача!



Обсуждение правил игры corewars выходит за рамки нашей статьи, но вы можете найти соответствующую документацию на странице man (man corewars).

Сборка corewars производилась по типичному сценарию. Существует множество вариантов использования опций скрипта *configure* для настройки включаемых в программу возможностей, использования различных команд из *Makefile* для настройки процесса компиляции и т.д.

Поскольку наша программа была установлена без использования *rpm*, она не попала в базу данных *rpm*. На случай, если установленная программа не работает, большинство *Makefile*'ов имеет параметр для деинсталляции, позволяющий удалить программу:

$make uninstall

Примите во внимание, что работа с сырым исходным кодом не вносит никаких изменения в базу данных RPM. Программами, установленными таким способом, будет невозможно управлять, так что будьте осторожны.

Исходные RPM'ы

При создании RPM используется термин исходные RPM (Source RPM). Это SPEC-файл в сочетании с исходным кодом, предназначенные для сборки на одной или нескольких архитектурах. Из двух возможных вариантов этот является лучшим. Используя исходные коды, вы можете скомпилировать программу на своей системе, но конечный продукт будет пригодным для установки пакетом RPM, а не сырым бинарником. Большая часть пакетов доступна как в виде предварительно

скомпилированных RPM, так и в виде SRPM. Это позволяет легко переносить программы между разными платформами в Linux. Если вы успешно пересобрали пакет для другой платформы, подумайте о том, чтобы сделать ваш конечный RPM доступным сообществу Linux.

Пусть исходники будут с вами

Если вы новичок в Linux, можете использовать другой способ установки программного обеспечения. Однако подход RPM к инсталляции изящен и предоставляет дополнительные возможности, которые вы быстро оцените.

Для работы с пакетами *rpm* из консоли вы должны хорошо знать, какие опции для чего используются, но для повседневного использования существуют варианты front-end интерфейсов, которые облегчают процесс управления пакетами *rpm*. Помимо тех, которые входят в ваш дистрибутив, доступны и другие, например, *Webmin*.

Вы не ограничены использованием только предварительно собранных пакетов. Вы можете воспользоваться преимуществами открытых исходников и собирать приложения непосредственно из исходников. В зрелых проектах сборка для как правило не вызывает затруднений. Помните, что программа, установленная из исходных кодов, не попадет в вашу базу данных *rpm*. Работая с исходниками, старайтесь использовать *source rpm*'ы, которые сочетают в себе возможность компиляции исходного кода и легкость управления пакетами rpm.

Особенности инсталляции в Debian

Дистрибутив Debian известен своей исключительной стабильностью и надежностью, а также замечательной системой управления пакетами/разрешения зависимостей *apt*. Установка новых приложений осуществляется очень просто:

 # apt-get программа

Выборка и установка библиотек, от которых зависит работа приложения, производятся автоматически. *apt* — изощренная, интеллектуальная программа, а официальный архив программного обеспечения Debian поддерживает жесткие стандарты качества пакетов. Официальный программный архив Debian содержит более 12 000 программ; это больше, чем у любой другой платформы.

dpkg —аналог RPM для Debian, обладающий дополнительными возможностями; он также выполняет базовую настройку конфигурации. Например, при установке Postfix *dpkg* запрашивает кое-какую информацию о системе, устанавливает стартовые и конфигурационные файлы и инициализирует программу.

Debian существует в трех разных версиях: стабильной, тестовой и нестабильной (также существует четвертая версия для искателей приключений — экспериментальная). Они обозначаются терминами Woody, Sarge и Sid. Версия Woody чрезвычайно консервативна. Пакеты допускаются в версию Woody только после обширной проверки зависимостей и исправления всех дефектов безопасности. Версии Sarge и Sid содержат новые пакеты, не прошедшие столь подробного тестирования.

«Заплатки» безопасности быстро выпускаются для Woody и весьма нерегулярно — для Sarge и Sid.

Какую версию использовать? Наиболее очевидный выбор — стабильная версия (Woody), надежная, как скала. Тем не менее, за надежность приходится расплачиваться: программы в Woody на месяцы, а иногда и на годы отстают от даты официального выпуска. Woody идеально подходит для серверов. Для настольных систем и рабочих станций более актуальны тестовая версия (Sarge) и нестабильная версия (Sid). И несмотря на устрашающие названия («тестовая», «нестабильная»), они работают вполне нормально.

Кодовые обозначения выглядят симпатично, но не стоит использовать их в конфигурационных файлах. Нестабильной версии всегда будет соответствовать обозначение Sid, но Woody и Sarge не всегда будут связываться со стабильной и тестовой версиями — когда-нибудь текущая тестовая версия Sarge будет повышена до статуса стабильной, а текущая версия Woody уйдет на покой. При грамотном сопровождении система Debian постоянно обновляется без переустановки, поэтому не стоит нарушать ее работу использованием кодовых обозначений, которые со временем изменятся.

Поиск программ для Debian

Вам понадобились программы для системы Debian. В Сети можно найти многие гигабайты программного обеспечения — но где найти программы, упакованные для Debian? И как выбрать архив?

Пакеты Debian устанавливаются из официальных архивов пакетов Debian, неофициальных архивов и с дисков CD-ROM. Источники указываются в файле /etc/apt/sources.list, после чего система управления пакетами Debian используется для установки пакетов из источников.

Для поиска отдельных пакетов можно воспользоваться страницей поиска Debian:

- http://www.debian.org/distrib/packages/.

Следующим шагом должно стать редактирование файла /etc/apt/sources.list и занесение в него выбранных источников.

При наличии нескольких источников apt-get всегда использует самую новую версию пакета. Список начинается с наиболее предпочтительных источников, поскольку apt-get обрабатывает список от начала к концу.

Редактирование файла sources.list — абсолютно законный, простой способ управления установкой программ. Занесите в файл все записи, которые вы когда-либо планируете использовать, и закомментируйте строки, не задействованные в конкретной установке.

В список sources.list рекомендуется включить официальные зеркала Debian, чтобы снять нагрузку с серверов Debian.org. Полный список официальных зеркал находится по адресу http://www.debian.org/mirror/.

Установка пакетов в системе на базе Debian

Все архивы программного обеспечения, компакт-диски и т. д. не принесут никакой пользы, если вы не умеете устанавливать программы. Итак, вы хотите знать, как установить новый пакет в Debian.

Воспользуйтесь командой *apt-get install*:

 # apt-get install tuxkart

Установка пакета с перезаписью файлов:

 # apt-get install --reinstall tuxkart

Чтобы установить сразу несколько программ, перечислите их, разделяя пробелами:

 # apt-get install tuxkart gltron frozen-bubble

tuxracer nethack galaga

Загрузка программ без установки или распаковки:

 # apt-get -d install tuxkart

Чтобы протестировать команду перед выполнением, присоедините к строке ключ *-dry-run*:

 # apt-get install tuxkart gltron frozen-bubble tuxracer nethack galaga --dry-run

Для определения имен пакетов используйте страницу поиска Debian по адресу http://www.debian.org/distrib/packages/. Имена пакетов Debian часто отличаются от своих аналогов из RPM. Например, программа CyrusSASL оформляется в пакет *sasl-2.x.rpm*, а в Debian она разбивается на несколько пакетов с именами *libsasl-**.

Не забудьте выполнить команду *apt-get update* после изменения */etc/apt/sources.list* и периодически запускайте ее, чтобы получать обновленную информацию из архивов пакетов.

apt-get загружает и устанавливает (а при необходимости и удаляет) все пакеты, необходимые для разрешения всех зависимостей.

УДАЛЕНИЕ ПАКЕТОВ ИЗ СИСТЕМЫ DEBIAN

Требуется удалить пакет или несколько пакетов из системы Debian.

Воспользуйтесь командой *apt-get remove:*

 # apt-get remove tuxpaint

Предварительное тестирование команды *remove:*

 # apt-get remove tuxpaint -dry-run

Удаление всех следов существования пакета, включая конфигурационные файлы:

 # apt-get --purge remove tuxpaint

Чтобы удалить сразу несколько программ, перечислите их, разделяя пробелами:

 # apt-get remove tuxkart gltron frozen-bubble tuxracer nethack galaga

Установка программ в Debian по исходным текстам

Требуется откомпилировать программу в системе (вместо установки двоичных файлов Debian). Возможно, вы хотите отредактировать код программы или запустить пакет из тестовой или нестабильной версии в стабильной (перекомпиляция обеспечивает построение правильных зависимостей).

Воспользуйтесь командой apt-get source и содействием dpkg.

Для начала загрузите заголовки и библиотеки, от которых зависит устанавливаемая программа, в каталог, в котором будет строиться пакет:

 # cd /usr/src

 # apt-get build-dep tuxkart

Загрузка и сборка пакета:

 # apt-get -b source tuxkart

Установка пакета:

 # dpkg -i tuxkart.deb

Построение пакетов .deb по исходным текстам чаще всего выполняется с целью использования пакетов из тестовой или нестабильной версии в стабильной. Перекомпиляция настраивает зависимости для стабильной версии.

Если потребуется установить программу, отсутствующую в архивах Debian, постройте .deb при помощи CheckInstall — утилиты, создающей пакеты RPM, .deb и Slackware.

Обновление пакетов в Debian

Требуется обновить пакет в системе Debian, потому что новая версия обладает расширенными возможностями или в ней исправлены ошибки.

Воспользуйтесь командой *apt-get install*:

 # apt-get install gltron

Обновление нескольких пакетов:

 # apt-get install tuxkart gltron frozen-bubble tuxracer nethack galaga

Команды обновления отдельных пакетов не существует. Команда apt-get install всегда устанавливает последнюю версию пакета.

Обновление системы Debian

Требуется обновить все пакеты в системе и заменить их новейшими версиями.

Убедитесь в том, что файл */etc/apt/sourcs.list* содержит ссылки на нужные источники, а затем выполните команду *apt-get upgrade*.

Всегда начинайте с обновления списков пакетов:

 # apt-get update

Следующая команда обновляет все установленные пакеты, но не удаляет пакеты для разрешения зависимостей:

 # apt-get -u upgrade

Обновление всех установленных пакетов с удалением или установкой пакетов по мере необходимости для разрешения всех зависимостей:

 # apt-get -u dist-upgrade

Флаг -u позволяет заранее просмотреть список всех изменений. Обновление может занять несколько часов, в зависимости от скорости подключения к Интернету и количества загружаемых пакетов.

Чтобы флаг -u применялся по умолчанию, отредактируйте (или создайте) файл /etc/apt/apt.conf:

// Всегда перечислять обновляемые пакеты

// и запрашивать подтверждение у пользователя

APT::Get::Show-Upgraded "true";

Перед каждым выполнением команды *apt_get dist_upgrade* сначала выполните команду *apt-get upgrade*, чтобы снизить вероятность ошибок при выполнении *dist-upgrade.*

Поиск программ, установленных в системе Debian

Требуется узнать, какие пакеты установлены в системе, к какому пакету относятся те или иные файлы и что находится в отдельных пакетах.

Воспользуйтесь средствами обработки запросов dpkg.

Следующая команда выводит список всех установленных пакетов и направляет его в файл:

$ dpkg -l | tee dpkglist

Поиск всех пакетов, удовлетворяющих критерию поиска, и вывод информации об их состоянии:

*$ dpkg -l '*gnome*'*

Поиск установленных пакетов, удовлетворяющих критерию поиска:

***$ dpkg -l | grep** gnome*

Вывод списка файлов, входящих в пакет:

$ dpkg -L gnome-applets

Операции с кэшем пакетов Debian

Требуется обеспечить актуальность кэша пакетов и списков пакетов, чтобы программа apt работала правильно, без возникновения ложных проблем с зависимостями.

Воспользуйтесь программами apt и dpkg.

Не забывайте выполнять команду apt-get update после модификации файла /etc/apt/sources.list, а также регулярно выполняйте ее, чтобы список пакетов оставался актуальным.

Следующая команда выводит список загруженных, но не установленных пакетов:

$ dpkg --yet-to-unpack

Проверка нарушенных зависимостей:

$ apt-get check

Удаление кэшированных пакетов, которые стали ненужными:

$ apt-cache autoclean

Удаление всех кэшированных пакетов:

$ apt-cache clean

Вывод списка частично установленных пакетов:

$ dpkg -audit

Если вызов dpkg --audit возвращает какие-либо результаты, как в следующем случае:

$ dpkg --audit

vpw (no information available)

для начала убедитесь в том, что возвращаемый пакет существует:

$ dpkg -l vpw

Package 'vpw' is not installed and no info is available

Если пакет существует, либо завершите установку, либо удалите ее. Если пакет не установлен, поищите соответствующую запись в */var/lib/dpkg/available* и */var/lib/dpkg/status* и удалите ее.

Разрешение конфликтов зависимостей в Debian

Программа не устанавливается из-за проблем с зависимостями, или команда *apt-get dist-upgrade* оставляет пакеты, которые нужно удалить.

Существует несколько команд, которые могут пригодиться для решения этой задачи; в этом разделе они перечисляются в том порядке, в котором их стоит попробовать.

Допустим, проблемы возникли с пакетом libpam-modules, который не желает обновляться:

 # apt-get install libpam-modules

...

The following packages have unmet dependencies:

libpam-modules: Depends: libdb3 (>=3.2.9-19.1) but 3.2.9-19 is to be installed

E: Broken packages

Если вы работаете в смешанной системе, для начала попробуйте указать версию:

 # apt-get install -t stable libpam-modules

Если это не помогает или система не является смешанной, попробуйте воспользоваться системой разрешения конфликтов Debian:

 # apt-get -f install

Затем выполните команду:

 # dpkg --configure -a

И снова повторите команду:

 # apt-get -f install

Если будет получен следующий ответ, значит, попытка завершилась неудачей:

Reading Package Lists... Done

Building Dependency Tree... Done

0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.

Теперь проверьте, что произойдет при удалении существующего пакета *libpam-modules*:

 # apt-get remove --dry-run libpam-modules

Reading Package Lists... Done

Building Dependency Tree... Done

The following packages will be REMOVED:

adduser adminmenu apache at base-config courier-imap courier-imap-ssl courier-pop courier-pop-ssl cron cupsys cupsys-driver-gimpprint dict-elements dict-foldoc dict-gcide dict-jargon dict-vera dict-wn dictd gdm2...

....

WARNING: The following essential packages will be removed

This should NOT be done unless you know exactly what you are doing!

login libpam-modules (due to login)

В данном случае исправление конфликтов зависимостей потребует едва ли не полной перестройки системы. Чаще проблемы ограничиваются несколькими пакетами. В этом случае начинайте удалять наименее важные из них, пока конфликты зависимостей не будут разрешены, а затем переустановите все пакеты, которые вам нужны.

Если apt-get -u dist-upgrage отображает какие-либо задержанные (held) пакеты, от них лучше избавиться. Пакеты удерживаются из-за конфликтов зависимостей, которые не удается разрешить apt. Попробуйте воспользоваться следующей командой для поиска и исправления конфликтов:

 # apt-get -o Debug::pkgProblemResolver=yes dist-upgrade

Если исправить конфликты не удается, команда завершается с сообщением

0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.

Удаляйте held-пакеты один за одним, каждый раз запуская *dist-upgrade*, пока не останется ни одного задержанного пакета. Затем установите заново все необходимые пакеты. Не забудьте использовать ключ *--dry-run*, чтобы заранее знать обо всех последствиях:

 # apt-get remove --dry-run libsdl-perl

В «монолитных» системах подобные конфликты версий встречаются редко. Пользователи смешанных систем сталкиваются с ними чаще. Чтобы предотвратить их, будьте чрезвычайно осмотрительны при выборе устанавливаемых программ — при первой установке Debian потратьте немного времени и выберите каждый необходимый пакет.

Список контрольных вопросов

1. Каковы основные принципы инсталляции программного обеспечения в Linux
2. Два основных подхода к инсталляции (Red Hat и Debian)
3. Какова структура RPM?
4. Графический интерфейс для инсталляции ПО
5. Порядок инсталляции из исходных кодов
6. Особенности инсталляции в Debian
7. Поиск программ для Debian
8. Установка программ в Debian по исходным кодам
9. Разрешение конфликтов зависимостей в Debian

Задания по лабораторной работе

1. Установить сервер OpenSSH с помощью утилиты *rpm*.
2. Проверьте текущее состояние установленных файлов пакета
3. Установите игрушку Corewars из исходных кодов
4. Удалите игрушку Corewars
5. Обновите список пакетов при помощи *apt*
6. Очистите установочный кэш

Список литературы

1. *Уэлш. М. и~др.,* Руководство по установке и использованию системы **Linux**. М.: ИЛКиРЛ, 1999
2. Александр Боковой, Александр Колотов, Александр Прокудин, Алексей Новодворский, Алексей Смирнов,Анатолий Якушин, Антон Бояршинов, Антон Ионов, Вадим Виниченко, Виталий Липатов, Георгий Курячий, Даниил Смирнов, Дмитрий Аленичев, Дмитрий Левин, Илья Трунин, Кирилл Маслинский, Максим Отставнов, Мэтт Уэлш, Олег Власенко, Сергей Турчин, Станислав Иевлев, Юрий Коновалов и другие; ALT Linux снаружи. ALT Linux изнутри, ISBN 5-9706-0029-6, Издатель: ДМК пресс, 2006 г. Москва
3. Марк Г. Собелл, Практическое руководство по Red Hat Linux: Fedora Core и Red Hat Enterprise Linux, 2-е издание (Practical Guide to Red Hat Linux: Fedora Core and Red Hat Enterprise Linux), 1072 стр., с ил.; ISBN 5-8459-0841-8, 0-13-147024-8; формат 70x100/16;твердый переплет DVD-ROM; 2005, 2 кв.; Вильямс.
4. Разработка приложений в среде Linux. Программирование для linux, 2-е издание, Майкл К. Джонсон, Эрик В. Троан
5. Руководство администратора Linux. Установка и настройка. 2-е издание, Эви Немет, Гарт Снайдер, Трент Хейн
6. Linux. Библия пользователя, Кристофер Негус
7. Linux для чайников , 6-е издание, Ди-Анн Лебланк
8. Разработка ядра Linux, 2-е издание, Роберт Лав
9. Библиотека Qt 4. Программирование прикладных приложений в среде Linux., Чеботарев Арсений Викторович
10. Red Hat Linux Fedora 4. Полное руководство, Пол Хадсон, Эндрю Хадсон, Билл Болл, Хойт Дафф
11. Искусство программирования для Unix, Эрик С. Реймонд
12. Linux для "чайников", 5-е издание, Ди-Анн Лебланк
13. Red Hat Linux. Секреты профессионала, Наба Баркакати
14. Использование Linux, Apache, MySQL и PHP для разработки Web-приложений, Джеймс Ли, Брент Уэр
15. Секреты хакеров. Безопасность сетей - готовые решения, 4-е издание, Стюарт Мак-Клар, Джоэл Скембрей, Джордж Курц
16. FreeBSD: полный справочник., Родерик Смит

17. Секреты хакеров. Безопасность Linux — готовые решения, 2-е издание, Брайан Хатч, Джеймс Ли, Джордж Курц
18. Red Hat Linux 8. Библия пользователя, Кристофер Негус
19. Серверы Linux. Самоучитель, Птицын Константин Александрович
20. Безопасность Linux, 2-е издание, Скотт Манн, Эллен Л. Митчелл, Митчелл Крелл
21. Сетевые средства Linux, Родерик Смит
22. Руководство администратора Linux, Эви Немет, Гарт Снайдер, Трент Хейн
23. Сети TCP/IP, том 3. Разработка приложений типа клиент/сервер для Linux/POSIX, Дуглас Камер, Дэвид Л. Стивенс
24. Секреты хакеров. Безопасность Linux — готовые решения, Брайан Хатч, Джеймс Ли, Джордж Курц
25. Программирование для Linux. Профессиональный подход, Марк Митчелл, Джеффри Оулдем, Алекс Самьюэл
26. Использование Linux, 6-е издание. Специальное издание, Дэвид Бендел, Роберт Нейпир
27. Создание сетевых приложений в среде Linux, Шон Уолтон
28. Освой самостоятельно Linux за 24 часа, 3-е издание,
29. Система электронной почты на основе Linux. Руководство администратора, Ричард Блам
30. Системное администрирование Linux, М. Карлинг, Стефан Деглер, Джеймс Деннис

Учебное издание

Инсталляция программного обеспечения в Linux

Методические указания

Составитель: Сухов Андрей Михайлович

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.