Глава 1

Общие сведения о программах Linux

- Особенности программ Linux
- Установка программ в Linux

Linux — это не программа, а операционная система со своими принципами построения и особенностями. Здесь все не так, как в Windows: например, в Linux нет привычных обозначений дисков C:, D:, любой компонент, в том числе и графическую среду, можно без последствий заменить альтернативным вариантом, и еще много другого, к чему придется привыкнуть. Одна из первых проблем, с которой столкнется любой пользователь, — порядок установки, обновления и удаления программ, в процессе чего выясняется, что скачать файл и дважды щелкнуть на нем кнопкой мыши в большинстве случаев недостаточно. Какой файл качать, что еще нужно сделать, чтобы программа установилась и, главное, работала? Обо всем по порядку.

Особенности программ Linux

Главный девиз UNIX-систем звучит как KISS — Keep It Simple Stupid, то есть «не усложняй». В качестве доказательства такой пример. У автора дома по Linux всего одна книга, которую он смог найти в магазинах в период, когда начал интересоваться системой. Это издание «Использование Linux» Джека Такета. Книге уже семь лет, но большая часть материала действительна до сих пор. В Интернете можно найти много старых книг, и все они в большинстве своем актуальны. Попробуйте применить книгу, написанную о Windows 98 или Windows XP к Vista, — почти весь материал окажется бесполезным.

Разработчик программы для Windows часто вынужден писать с нуля большинство функций, так как все защищено патентами, и либо платишь, либо делаешь все сам. На это уходит время, разработчику приходится самому разбираться в множестве вопросов либо привлекать людей, которые помогут добавить нужную функциональность, что приводит к удорожанию продукта. Зато программа устанавливается двойным щелчком мыши, так как все компоненты уже включены в дистрибутив. В Linux процесс выглядит несколько иначе.

ПРИМЕЧАНИЕ

GNU — это рекурсивная аббревиатура от GNU's Not UNIX (GNU — не UNIX).

Программы, библиотеки и прочее распространяются с исходными кодами под лицензией GNU GPL (General Public License — стандартная общественная лицензия), которая ставит только одно условие: если при разработке используется программа с этой лицензией, то и полученный продукт также должен распространяться по лицензии GNU GPL.

ПРИМЕЧАНИЕ

Слово free («свободный») в английском языке имеет несколько значений, в том числе и «бесплатный».

На сайте проекта GNU (http://www.gnu.org/home.ru.html) дано пояснение английского слова free. По их мнению, программное обеспечение — это предмет свободы, а не цены, и, чтобы понять идею, свободное программное обеспечение не нужно ассоциировать с бесплатным мороженым, то есть свободное не обязательно должно быть бесплатным. Свободное программное обеспечение позволяет пользователям свободно запускать, копировать, распространять, изучать, изменять и улучшать программное обеспечение. Более точно это раскрыто в четырех типах свободы для пользователей программного обеспечения:

- свободно запускать программы для любых целей (свобода 0);
- свободно изучать, как работает программа, и иметь возможность свободно адаптировать ее под свои нужды (свобода 1); доступ к исходным кодам является непременным условием;
- свободно распространять копии, чтобы помочь тем, кто интересуется свободным программным обеспечением (свобода 2);
- свободно улучшать программы и делать улучшения доступными для других; из этого извлекает пользу все сообщество (свобода 3); доступ к исходным кодам является непременным условием.

Такой подход дает программисту простоту и быстроту разработки. Он использует уже готовые компоненты, не вникая в суть их работы, и создает новый продукт. Каждый улучшает только свою часть продукта. Так, в Linux очень много консольных утилит. Начинающему пользователю непросто освоить все параметры командной строки, ему удобнее, чтобы программа имела понятный и легкий в освоении графический интерфейс. Программист же не создает новый продукт, он берет все консольные утилиты и пишет к ним графическую надстройку — фронт-энд (front-end), который скрывает от пользователя особенности работы консольных программ, часто добавляя новую функциональность. Например, популярный консольный видеопроигрыватель MPlayer имеет несколько таких надстроек — Kplayer, KMplayer, smplayer, Freevo и другие.

Однако чтобы такая программа заработала, необходимо установить все остальные компоненты и библиотеки, которые она использует. Например, для популярной утилиты записи дисков K3B потребуется установить 48 библиотек и программ (рис. 1.1).

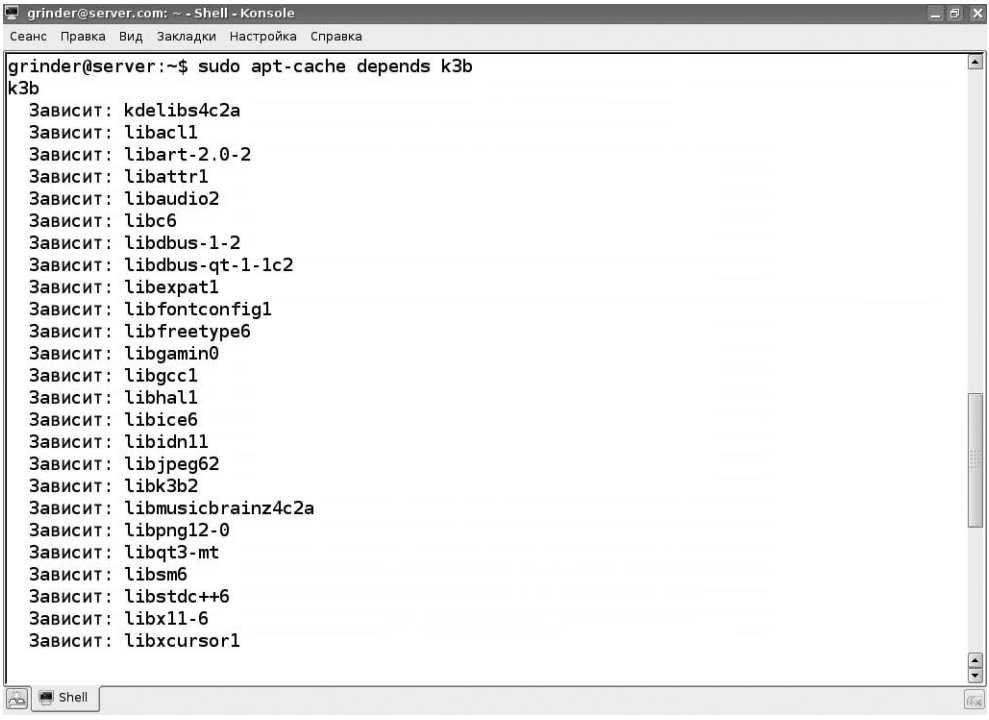

Рис. 1.1. Список зависимостей пакета K3B

Такие пакеты называются зависимостями. Большинство из них, вероятно, уже есть в системе, но если не будет хотя бы одного, то установить программу будет проблематично либо после установки она не будет иметь полную функциональность. Зависимости бывают двух видов:

- жесткая (hard): обязательная, без нее программа, скорее всего, не будет работать;
- мягкая (soft): рекомендуемая, без нее программа работать будет, но, установив такой пакет, можно добавить приложению функциональность (например, работу с некоторым форматом файлов).

Именно поэтому, скачав всего один пакет, установить программу часто не получается.

Есть еще одна проблема, о которой необходимо знать. В отличие от Windows, релизов которой не так много, дистрибутивов Linux огромное количество. Они отличаются не только названием, так как Linux — это только ядро, разрабатываемое под руководством Линуса Торвальдса, а все остальное — это что-то вроде набора «Сделай сам». Эти наборы хороши, но часто не только не совместимы между

собой, но и для каждой новой версии одного и того же дистрибутива потребуется специально собранный пакет. Не будем вдаваться в подробности, тем более что не все так плохо, так как в популярных решениях используются пакетные системы, скрывающие такие особенности, но об этом просто нужно знать.

Установка программ в Linux

Изначально Линус Торвальдс предоставлял пользователям только ядро, а остальное они компилировали сами. Сейчас ситуация полностью изменилась, и в современных дистрибутивах для установки программы достаточно одного щелчка кнопкой мыши.

Компиляция программ

Даже после появления пакетов, которые представляли собой уже скомпилированные программы, компиляция долгое время оставалась и для некоторых остается основным средством установки.

ПРИМЕЧАНИЕ

Первые прекомпилированные наборы появились в дистрибутиве SLS Linux, который в начале 90-х годов был самым популярным дистрибутивом. Его создатели сочли неудобным самостоятельнную сборку приложений, поэтому пользователям были предложены не только само ядро Linux и основные утилиты, но и набор разнообразного программного обеспечения с простой программой установки. Все это поставлялось уже в скомпилированном виде, исходный текст прилагался только для самых основных компонентов вроде ядра. Кстати, основой популярных сегодня дистрибутивов Slackware и Debian послужил именно SLS.

Несмотря на обилие пакетных систем, о которых пойдет речь далее, многие пользователи предпочитают самостоятельно собирать программы, так как в репозиториях часто находится устаревшая версия программ, и приходится ждать, когда сборщик займется ею.

ПРИМЕЧАНИЕ

Человека, который отвечает за сборку пакета, называют maintainer, в Интернете часто можно встретить это слово в русской транскрипции — майнтейнер, или сборщик. Он отвечает за то, что пакет протестирован, работоспособен и его установка не вызовет сложностей. В процессе сборки он может использовать патчи (англ. patch — «заплатка»), изменяющие функциональность или устраняющие найденные ошибки. Однако таких людей мало, и все пакеты отслеживать трудно, поэтому проекты постоянно привлекают добровольцев.

Кроме этого, иногда требуется изменить функциональность программы при помощи патчей, не используемых майнтейнером. Последняя причина — это производительность: чтобы пакет работал на максимально большом количестве оборудования, его собирают с оптимизацией под определенный тип процессора — как правило, это i386, то есть под процессор Intel 80386, выпущенный еще в 1985 году. Встречаются сборки i486, i586 (Pentium MMX) и i686 (Pentium Pro), в последнее время появились сборки под 64-битные процессоры. Собирая программу самостоятельно, пользователь может скомпилировать ее под свой тип процессора, под мультипроцессорные системы и прочее. Эксперименты доказывают, что производительность получаемого пакета увеличивается (в зависимости от аппаратного обеспечения) от 10 до 200 % по сравнению с бинарными сборками, поставляемыми с дистрибутивом, и чем новее оборудование, тем больше эффективность такой сборки, хотя она не всегда нужна.

Установка программ, распространяющихся в виде исходных файлов, вызывает трудности у большинства начинающих пользователей Linux, хотя считается, что это один из основных навыков, необходимых для работы. Это несколько устаревшее мнение, но попробуем разобраться.

ВНИМАНИЕ

Для самостоятельной компиляции программ потребуются компилятор GCC, GNU Make и прочие утилиты. В некоторых дистрибутивах они по умолчанию не устанавливаются. И чтобы установить их, в Kubuntu необходимо выполнить команду sudo apt-get install build-essential.

Как правило, исходные файлы заархивированы в так называемый тарболл (tarball). Архив, в зависимости от программы, которая использовалась при его создании, имеет суффикс `.tar.gz` или `.tar.bz2` и имя типа `program-x.xx`, где `program` — это название программы, а `x.xx` — версия.

В общем виде процесс установки выглядит так. Сначала распаковывается архив с помощью графической программы или команд:

```
tar xzvf program-x.xx.tar.gz
tar xjvf program-x.xx.tar.bz2
```

Затем следует перейти в образовавшийся каталог. В нем можно найти файлы с именами `INSTALL` или `README`, в которых кратко описан процесс установки, сказано о зависимостях программы, даны описание программы и инструкции по сборке. В общих случаях достаточно выполнить команду `./configure` без дополнительных параметров. Сценарий самостоятельно найдет все программы и библиотеки

и, если все в порядке, создаст makefile, необходимый для компиляции программы. Однако в случае, если сценарий что-то не найдет, последует сообщение об ошибке — внимательно прочитайте его и постарайтесь понять, в чем дело.

ВНИМАНИЕ

В дальнейшем при описании консольных команд будет использоваться знак доллара ($), что означает, что для выполнения этой команды достаточно прав обычного пользователя, или знак решетки (#), что требует прав root.

Скорее всего, в сообщении будет сказано, что отсутствует какой-то файл или библиотека либо имеющаяся версия устарела. В этом случае придется искать и устанавливать недостающее. Однозначного совета дать невозможно, здесь потребуется некоторый опыт. В случае если сообщение об ошибке непонятное, лучше поискать ответ на различных форумах — велика вероятность, что кто-то уже столкнулся с похожей проблемой и нашел ее удачное решение. Задавая вопрос, вставьте в него текст сообщения и укажите название и версию устанавливаемой программы и используемого дистрибутива. Для получения справки о дополнительных параметрах сценария следует использовать ключ `--help`.

```
$ ./configure --help
```

Наиболее часто используемым параметром является `--prefix`, с помощью которого указывается отличный от используемого по умолчанию каталог для установки программ (обычно `/usr/local/`). В некоторых случаях такого сценария нет, а есть уже готовый makefile. В этом случае сразу переходите ко второму этапу — выполните команду `make`, которая скомпилирует программу; если работа `make` завершилась без ошибок, устанавливайте приложение (при этом устанавливаются не только двоичные файлы, но и документация):

```
# make install
```

Для выполнения последнего шага понадобятся права суперпользователя (root), которые можно получить, выполнив команду `su` или `sudo`:

```
$ sudo make install
```

В дистрибутиве Kubuntu, чтобы пользователь мог выполнить команду `sudo`, он должен входить в группу admin. Первый созданный при установке пользователь заносится в нее автоматически. Для удаления установленной таким образом программы следует использовать команду `make uninstall`, хотя часто разработчики не используют в сценариях эту возможность.