Практическая работа 2. Пакеты. Пакетные менеджеры.

Пакетный менеджер (или система управления пакетами) — набор программного обеспечения, позволяющего управлять процессом установки, удаления, настройки и обновления различных компонентов программного обеспечения.

В различных дистрибутивах Linux используются разные пакетные менеджеры:

- Ubuntu/Debian и все дистрибутивы на их основе - apt RHEL/CentOS/Fedora - yum (устаревший), dnf arch и все дистрибутивы на его основе - pacman

Практическая работа будет рассмотрена на примере пакетного менеджера apt.

Репозиторий — это все файлы пакетов, принадлежащие одному дистрибутиву (например, Fedora), то есть огромное хранилище пакетов, которое находится в сети Интернет и которым Вы можете бесплатно воспользоваться. Те самые ISO-файлы образов для записывания на болванку и последующей установки содержат как раз репозитории пакетов со всеми зависимостями и менеджером пакетов плюс установочную программу, которая разметит жёсткий диск, всё поставит и приготовит Вам рабочий стол (или сервер, или что попросите).
Они создаются для централизованного управления обновлением пакетов.

Под **пакетами** в Linux подразумевается программное обеспечение (ПО), которое Вы хотите установить на компьютер. Скажем, например, в Windows софт устанавливается с помощью мастера (программы) установки — setup.exe или install.exe. Вы запускаете этот мифический экзешный файл, и процесс установки начинается едва ли не мгновенно после выбора пути и мелких побочных настроек.

2.1 Работа с пакетным менеджером apt

APT (Advanced Package Tool- инструмент командной строки для взаимодействия с системой управления пакетами.

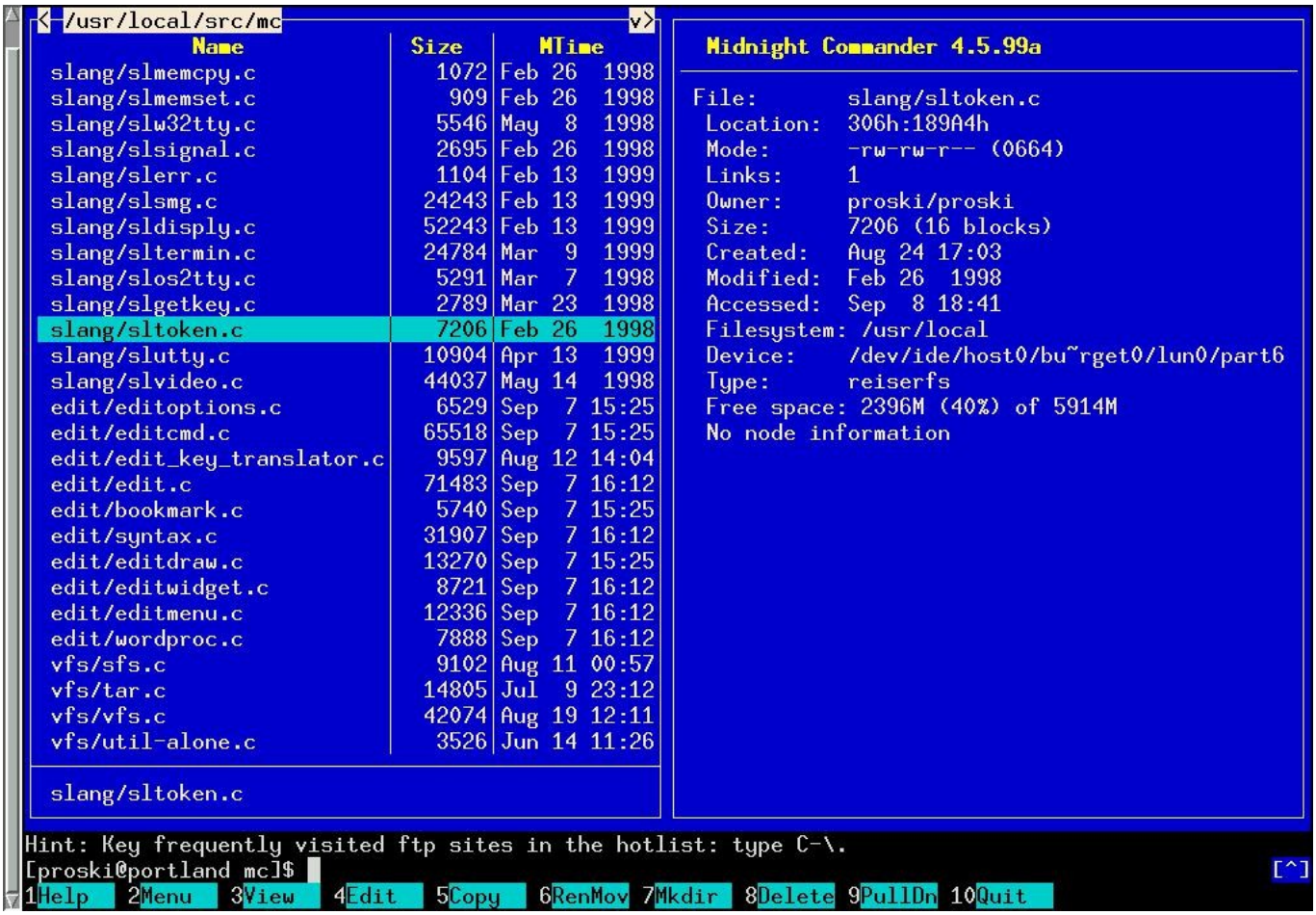
Использование:

- `apt update` - обновление репозиториев
- `apt upgrade` - обновление пакетов (выполняется после apt update)
- `apt install` - установка пакетов
- `apt remove` - удаление пакетов
- `apt autoremove` - удаление более ненужных пакетов (например, какие-то пакеты устанавливались как зависимость, и более стал не нужен), пользовательские конфиги не затрагиваются
  `apt purge` - удаление со всеми конфигурационными файлами `add-`
  В чем разница между apt remove и apt purge?
  apt remove просто удаляет двоичные файлы пакета, но оставляет файлы конфигурации.
  apt purge удаляет все, что связано с пакетом, включая файлы конфигурации.
- `apt-repository` - добавление репозитория в список используемых
- `apt-cache search` - поиск пакета по кешу синхронизированных
- репозиториев
- `apt search` - поиск пакета
- `apt show` - вывод информации о пакете
  `apt --help` - вывод краткой информации о пакете и основных сценариях использования
-

2.1.1 Установка нового пакета:

```
# apt install mc
```

После этого будет установлена утилита midnight commander (Аналог windows FAR Manager)



2.2 Добавление репозиториев

Рассмотрим на примере postgresql. Для того, чтобы добавить репозиторий, который ведет команда

PostrgeSQL, необходимо выполнить следующие команды:

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/
`lsb_release - cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'

$ wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo
apt-key add -

$ sudo apt-get update

$ sudo apt install postgresql postgresql-contrib
```

Что содержится в командах:

1. Добавляется репозиторий в список репозиториев apt (добавление строчки в sources.list)
2. Загружается ключ репозитория и добавляется в пакетный менеджер
3. Обновляется список пакетов
4. Установка пакета

В общем случае может быть более простой способ: `$ sudo add-apt repository ppa:репозиторий/ppa`

```
sudo apt-add-repository ppa:ripps818/coreavc
Персональные архивы пакетов (PPA)
```

2.3 Сборка пакетов из исходных кодов

В общем случае сборка пакетов выглядит следующим образом:

1. Загрузка архива с исходным кодом (чаще всего .tar.gz)
2. Разархивирование (либо в ui, либо см. tar и его использование)
3. Переход в директорию
4. `./configure`
5. `make`
6. `sudo make install`

Загрузите архив sqlite и установите его в систему на основе вышеперечисленных действий

2.4 Создание собственного deb пакета

Подготовка окружения

0. Подготовьте или соберите программу/скрипт, которую собираетесь пакетировать
1. Установите необходимые пакеты для сборки пакетов sudo apt-get install dpkg debconf debhelper lintian
2. Для начала необходимо создать директорию, в которой будем проводить свои манипуляции, а также все необходимые директории, которые будут использоваться при сборке и установке пакета

```
mkdir ~/package_name

mkdir -p ~/package_name/DEBIAN # управляющая папка
mkdir -p ~/package_name/usr/bin # путь к скрипту или исполняемому файлу
cp binary_file ~/package_name/usr/bin/ # копируем наш скрипт или
программу в нужное место
```

В итоге имеем: package_name/DEBIAN/ package_name /usr/ package_name/usr/bin/

Описание пакетирования в debian

Директория DEBIAN содержит в себе как минимум 1 файл, который описывает пакет (control), остальные используются либо для прикрепления текстовой информации (changelog, лицензия), либо

для управления расширенными возможностями установки приложений.

control — центральный файл пакета, описывающего все основные свойства. Файл — текстовый, состоящий из пар «Атрибут: значение». Можно использовать комментарии: символ "#" в начале строки (возможность была добавлена в версии dpkg >= 1.10.11).

Атрибут	**Описание**	**Примеры**
-	Основные	
Package:	Имя пакета: [a-zA-Z0-9-] — только латиница, цифры, и дефис. Имя используется при установке: apt-get install <package>	Package: supersh
Version:	Версия пакета (и проги внутри). Используется для определения «обновлять ли». Формат принят такой: <версия_программы>- <версия_пакета>. Рекомендую всегда указывать версию пакета: при изменении структуры пакета цифра увеличивается на единичку. Допустимые символы достаточно вольные: можно использовать дату и буквы.	Version: 2009.12.12-1
Provides	Имя приложения (возможно, виртуальное), регистрируемое в системе в результате установки этого пакета. Используется редко: в основном, если нужно изменить имя пакета, или если более одного пакета предлагают одинаковый функционал. Например, пакеты Apache и nginx предоставляют возможность демона httpd: Provides: httpd. Вы наверняка сталкивались с ошибкой при попытке установки: «is a virtual package». Это оно и есть.	Provides: package_name
Maintainer	Имя и почта мэйнтейнера пакета: человека, который опакетил приложение.Формат произвольный, но принято имя <e-mail>	Maintainer: user user@doamin.com
Architecture	Архитектура процессора, для которой предназначен пакет. Допустимые значения: i386, amd64, all, source. all используется для скриптов. source используется для компилируемых пакетов с исходниками	Architecture: all
Section	Определяет задачу, для которой приложение обычно используется (группа приложений). Возможные значения: admin, base, comm, contrib, devel, doc, editors, electronics, embedded, games, gnome, graphics, hamradio, interpreters, kde, libs, libdevel, mail, math, misc, net, news, non-free, oldlibs, otherosfs, perl, python, science, shells, sound, tex, text,	Section: misc

	utils, web, x11	
Description	Описание пакета. Описание состоит из двух частей: короткое описание (70 символов) на той же строке, и длинное описание на последующих строках, начинающихся с пробела. В расширенном описании все переводы строки игнорируются. Для вставки \n используется одиночная точка.	Description: Short. ␣Long ␣goes here. ␣. ␣New line.

Атрибут	**Описание**	**Примеры**
—связи и зависимости		
Depends	Список пакетов через запятую, которые требуются для установки этого пакета. После имени пакета можно в круглых скобках указать ограничение на версию, используя операторы: <<, =, >>, <=, >=. Если оператор не указан — используется >=	Depends: dpkg, libz (>= 1.2.3), jpeg (= 6b), png (< 2.0)
Pre-Depends	Список пакетов, которые требуются в процессе установки этого пакета.Эти зависимости могут потребоваться для скриптов установки пакета: например, пакет flash-installer требует wget. Можно использовать ограничения на версию (см. Depends).	Pre-Depends: wget (>= 1.0)
Conflicts	Список пакетов, которые не могут быть установлены одновременно с этим. Установка не удастся, если хоть один из перечисленных пакетов уже будет установлен.	Conflicts: opencv
Replaces	Список пакетов, файлы которых модифицируются этим пакетом. Требуется в случае создания «пакета-патча», изменяющего что-либо: в противном случае при замене файлов чужого пакета возникнет ошибка при установке.	Replaces: ut2004
Recommends	Список пакетов, рекомендуемых к установке Эти пакеты не обязательны, но обычно используются вместе с текущим	Recommends: superplatform
Suggests	Список пакетов, предлагаемых к установке. По идее, менеджер пакетов должен предлагать установить их.	Suggests: supersh-modules
Build-Depends	Только для Architecture: source. Список пакетов, требуемых для компиляции исходников. То же, что и Depends, но логически отделено.	Build-Depends: cmake
—Дополнительное Размер файлов пакета в килобайтах.		
Installed-Size	Просто цифра, округлённая до ближайшего целого. Используется менеджером пакетов для определения суммарного требуемого объёма на	Installed-Size: 3

	диске.	
Priority	Приоритет пакета: насколько он важен в системе. Возможные значения: extra, optional, standard, important, required (такие пакеты не удаляются вообще!).	Priority: optional
Esssential	Если установить этот атрибут в значение «yes», пакет нельзя будет удалить.	Esssential: yes
Origin	Строка: откуда получены программы в пакете. Обычно используется URL сайта автора, почта или имя.	Origin: acme.com
X-Source	Полная ссылка на *.tar.gz архив с исходниками	X-Source: ...*.tgz

`DEBIAN/copyright` - © / лицензия Текст лицензии. Файл не обязателен

- `DEBIAN/changelog` - история изменений

  Changelog в специальном формате: используется dpkg для получения номера версии, ревизии, дистрибутива и важности пакета. Лучше посмотреть в официальной документации. Пример:

  ```
  package_name (1.0-1) stable; urgency=medium
  * Testing.
  ```

- `DEBIAN/rules` - правила компиляции

  Используется для управления компиляцией пакета. См. официальную документацию

- DEBIAN/conffiles: список файлов конфигурации Обычно пакеты содержат болванки конфигурационных файлов, например, размещаемых в /etc. Очевидно, что если конфиг в пакете обновляется, пользователь потеряет свой отредактированный конфиг. Эта проблема легко решается использованием папок типа «config.d», содержимое которых включается в основной конфиг, заменяя собой повторяющиеся опции. Файл «DEBIAN/conffiles» позволяет решить проблему иначе: он содержит список файлов конфигурации (по одному на строке). Если в текущей версии пакета один из этих файлов обновляется, то пользователь получает предупреждение о конфликте версий конфигов, и может выбрать: удалить, заменить, или сделать merge. На каждой строке должен быть полный абсолютный путь до каждого конфига. Например:

  /etc/supersh/init.conf

  /etc/supersh/actions.conf

- DEBIAN/dirs: список папок для создания «Список абсолютных путей к папкам, которые требуются программе, но по каким-либо причинам не создаются.» — гласит официальная документация. На практике – здесь перечисляются все папки, так или иначе используемые программой: и где лежат бинарники, и которые используются программой. По одной на строке. Например:

  /var/log/supersh

  /var/lib/supersh

  Удобно использовать для создания нескольких пустых папок.

Скриптинг

Скрипты позволяют управлять установкой, переустановкой и удалением пакета, выполняя действия, которые нельзя сделать простым копированием файлов в правильные места. Это может быть скачивание дополнительных файлов (как это делает flash-installer), изменение существующих, а также — вывод интерактивных (GUI или ncurses) диалогов, позволяющих пользователю сконфигурировать пакет под себя: например, mysql спрашивает какой установить пароль для root. Все скрипты выполняются от пользователя root. Также они получают аргументы (которые обрабатывать не обязательно), конкретизирующие на каком именно этапе находится установка. Подробнее об этом здесь.

DEBIAN/(preinst|postinst|prerm|postrm): скрипты

установки Всего можно создать до четырёх скриптов в

одном пакете:

Скрипт	**Назначение**
DEBIAN/preinst	Выполняется перед установкой пакета: он может подготовить что-либо для успешной установки
DEBIAN/ postinst	Выполняется сразу после установки пакета: он настраивает установленный пакет так, чтоб он был готов к работе. Здесь также выполняется интерактивная конфигурация пакета: это делается при помощи dh_input и файла DEBIAN/templates
DEBIAN/prerm	Выполняется непосредственно перед удалением пакета: обычно этот скрипт подчищает установочные пути пакета так, чтоб ничего лишнего не осталось
DEBIAN/postrm	Выполняется сразу после удаления пакета: вычищает остатки

Обратите внимание, что ошибки, возникающие в этих скриптах никак не логируются: ничего интереснее кода возврата скрипта нигде не сохраняется, и логирование необходимо делать вручную!

Сборка пакета

Первое, что нужно сделать — это рекурсивно выставить всем файлам в корне пакета пользователя и группу root:root (или другие, если потребуется). Это нужно затем, что файлы пакета упаковываются в tar.gz архив который сохраняет и права доступа к файлам, и владельца. Потому нужно выполнить:

```
$ sudo chown -R root:root
```

Однако делать это не обязательно. Есть отличная команда fakeroot которая при создании архива подменит владельца файлос root-ом. В нашем примере, скрипт должен иметь бит выполнимости. Потом выходим на папку назад, чтоб было видно корневую папку пакета, и пакет создаётся сам:

```
$ fakeroot dpkg-deb --build supersh
```

Созданный пакет необходимо переименовать, чтобы он соответствовал порядку именования *.deb пакетов: <имя пакета>*<версия>*<архитектура>.deb

```
$ mv supersh.deb supersh_1.0-1_all.deb
```

Автоматическая проверка пакета

Существует утилита lintian, позволяющая проверить пакет и выявить типичные ошибки в его структуре.
Делается это так:

```
$ lintian supersh_1.0-1_all.deb
```

Установка пакета

```
$ sudo dpkg -i supersh_1.0-1_all.deb
```

2.5 Создание и настройка локального deb-репозитория

Теперь у нас есть собственный пакет. Когда их будет несколько, и тем более — с зависимостями, окажется, что намного удобнее быстренько поднять собственный локальный микро-репозиторий, и включить его в список источников менеджера пакетов. Сперва установим помощника:

```
$ sudo apt-get install reprepro
```

Описание будущего репозитория

Центр репозитория — его описание. Главное в нём — список компонент репозитория. Мы создадим компоненты «soft» и «games». Выберите папку для будущего репозитория. Все действия производятся из её корня. Создаём файл conf/distributions следующего содержания:

```
Description: my local repository
Origin: Ubuntu
Suite: testing
AlsoAcceptFor: unstable experimental
Codename: karmic
Version: 5.0
Architectures: i386 amd64 source
Components: soft games
UDebComponents: soft games
```

Создание репозитория

Теперь сгенерируем шаблон на основе описания. Команды выполняются в корне репозитория:

```
$ reprepro export
$ reprepro createsymlinks
И добавим готовый репозиторий в
/etc/apt/sources.list: deb file:///path/to/repo/
karmic soft games
```

Этот репозиторий можно также расшарить при помощи веб-сервера.

Управление пакетами в репозитории

В корень репозитория кладём *.deb файлы для добавления, и добавляем их в компоненту soft дистрибутива karmic:

```
reprepro -C soft includedeb karmic *.deb
```

теперь пакеты доступны из менеджера пакетов. Удаление пакетов:

```
reprepro -C soft remove karmic supersh
```