Лекция 4. Установка пакетов, репозитории. Планировщик cron

Цель лекции

- Научиться устанавливать ПО из репозиториев
- Изучить процесс подключения дополнительных репозиториев
- Понять отличия deb и snap-пакетов
- Познакомиться с планировщиком задач cron

Термины

Репозиторий — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.
Пакет — под пакетами в Linux подразумевается программное обеспечение, которое можно установить, то есть набор файлов, объединенных для выполнения определённого функционала. Пакеты как правило хранятся в репозиториях.
PPA (сокр. от англ. Personal Packages Archive) — персональный архив пакетов. В отличие от других репозиториев Ubuntu, PPA-репозиторий содержит версии только какой-то одной программы.
apt — программа для установки, обновления и удаления программных пакетов в операционных системах Debian и основанных на них. В Apt есть коровья суперсила.
dpkg — это пакетный менеджер для Debian-систем. Он может устанавливать, удалять и создавать пакеты, но, в отличие от других систем управления пакетами, не может автоматически загружать и устанавливать пакеты или их зависимости.
Snap — это пакет, который, помимо готовой сборки самого приложения, включает в себя все необходимые зависимости и может работать почти в любом дистрибутиве Linux.
Планировщик заданий — программа (служба или демон), часто называемая сервисом операционной системы, которая запускает другие программы в зависимости от различных критериев, таких как, например, наступление определенного времени.

План

1. Варианты установки софта в Linux.
2. Пакетные менеджеры и репозитории.
3. Классификация репозиториев.
4. Подключение дополнительного репозитория.
5. Поиск и установка deb-пакетов.
6. Утилиты apt и dpkg.
7. Альтернативная система snap-пакетов.
8. Планировщик cron.
9. Системные файлы конфигурации crontab.

10. Пользовательские crontab.

Текст лекции

Варианты установки софта в Linux: ручная сборка, готовые сборки, пакеты

Существует множество способов установки программного обеспечения в Linux. Начнём с самого старого и сложного – **установка ПО из исходного кода**. В этом случае нам нужно сначала подготовить окружение, средства разработки, установить все нужные зависимости (библиотеки и утилиты), после скомпилировать (собрать) программу из исходников (обычно `make`) и затем отдельной командой поставить (например, `make install`). Такой метод требует множество ручной работы, подвержен ошибкам и вызывает большие сложности при регулярном обновлении ПО. Кроме того, ручная установка из исходников приводит к замусориванию системы (нет централизованного управления ПО). Этот метод сегодня используется редко, в основном для экспериментов и разработки системного софта.
Второй вариант – получение **скомпилированного софта** в виде архива исполняемых файлов. Здесь нет операции сборки и подготовки, но проблема с отсутствием централизованного управления также остаётся. Также используется в редких ситуациях отсутствия альтернатив.
Третий вариант – **установка ПО из пакетов**. Это основной метод в большинстве случаев. Пакет – это готовый набор файлов для установки софта в систему. Помимо файлов пакет содержит метаинформацию о зависимостях софта, конфликтах с другими пакетами, версии и так далее. Благодаря этим метаданным можно организовать централизованное управление пакетами – данные об установленных пакетах хранятся в локальной базе данных, а сами пакеты хранятся в репозиториях – специальных хранилищах. При использовании пакетов из репозиторием мы получаем быструю и удобную установку в одну команду (например, `apt install`), автоматизированное обновление всех пакетов (то есть, всей системы), простое удаление софта.
Давайте подробнее разберемся с тем, какие бывают пакетные менеджеры и репозитории.

Пакетные менеджеры и репозитории

В системе Ubuntu мы можем использовать пакеты двух типов: `deb` и `snap`.
Пакеты типа **deb** являются традиционными и проверенными временем. Они работают по принципу единой системы пакетов, которые образуют весь набор используемого ПО в операционной системе, включая компоненты самой ОС. Например, ядро ОС поставляется в виде пакета, все библиотеки, утилиты и их конфигурация также упакована в пакеты. При этом, пакеты могут зависеть друг от друга и конфликтовать. При установке пакета сначала будут установлены его зависимости и только потом он сам. Некоторые пакеты могут входить в зависимости сотен других пакетов (например, библиотеки). Конфликты могут возникать из-за принципиально различных версий софта, которые не могут работать рядом, или для ограничения установки одного экземпляра ПО в системе. Важно, что все репозитории, которые подключены к системе

должны поддерживать иерархию пакетов ОС, так как все репозитории работают в едином пространстве имён пакетов, зависимостей и конфликтов. Из-за этого ограничения могут возникать проблемы совместимости сторонних репозиториев с конкретными версиями операционных систем, что усложняет для разработчиков софта поддержку таких репозиториев.

Пакеты типа **snap** работают по другому принципу. Они содержат в себе все зависимости ПО и не связаны ограничениями deb-пакетов. ПО, установленное из snap-пакета работает в изоляции от основной системы и использует только свои библиотеки. В результате, размер snap-пакетов значительно выше, скорость работы и эффективность использование оперативной памяти – ниже. В достоинства snap можно записать высокую совместимость и простоту поддержки пакетов для разработчика софта.

Для серверного применение deb-пакеты остаются основным выбором, а snap-пакеты могут быть полезны для настольной версии (в случае отсутствия deb-пакета).

Пакеты хранятся специально организованных репозиториях. Далее подробнее разберём их особенности и классификацию.

Классификация репозиториев

Обновления и программное обеспечение в Linux-системах устанавливаются из репозиториев. **Репозиторий** — это хранилище пакетов, то есть файлов и библиотек, которые мы можем установить в ОС. Репозиторий может быть размещён локально, может находиться на носителе (флешке, DVD-диске), но чаще всего он размещён в интернете. Условно репозитории можно разделить на три группы:

1. **Стандартные репозитории** — это репозитории, поддерживаемые разработчиками операционных систем. Включают в себя стабильные версии программного обеспечения. Зачастую эти версии отстают на несколько шагов от последних версий пакетов.
2. **Дополнительные репозитории** — репозитории, поддерживаемые разработчиками программного обеспечения. Включают в себя последние стабильные версии ПО. Зачастую узкоспециализированы под конкретный пакет и библиотеки, необходимые для этого пакета.
3. **Неофициальные репозитории** — репозитории, созданные сообществом или одним человеком. Могут содержать в себе как последние стабильные, так и тестируемые версии программного обеспечения.

Программное обеспечение в Ubuntu делится на четыре вида по типу лицензирования и уровню поддержки:

1. **Main** — свободное ПО, официально поддерживаемое компанией Canonical.
2. **Restricted** — проприетарное ПО (в основном драйверы устройств), официально поддерживаемое компанией Canonical.
3. **Universe** — свободное ПО, официально не поддерживаемое компанией Canonical, но поддерживаемое сообществом пользователей.
4. **Multiverse** — проприетарное ПО, не поддерживаемое компанией Canonical.

Официальные репозитории Ubuntu делятся на следующие типы (по версиям ПО):

1. $release — пакеты на момент выхода релиза.
2. $release-security — пакеты критических обновлений безопасности.

3. $release-updates — пакеты обновления системы, то есть более поздние версии ПО, вышедшие уже после релиза.
4. $release-backports — пакеты более новых версий ПО, которое доступно только в нестабильных версиях Ubuntu.
5. partner — репозиторий, содержащий ПО компаний-партнёров Canonical.

Информация о подключённых репозиториях в Ubuntu хранится в каталоге `/etc/apt/`, в файле `sources.list`. Репозитории защищают от подмены при помощи сверки цифровых подписей репозитория и клиента. В репозитории хранится закрытая часть ключа, у клиента — открытая часть ключа.

Подключение дополнительного репозитория

В Ubuntu репозитории можно подключить тремя способами: используя графический интерфейс, путём редактирования файла `/etc/apt/source.list` (или созданием файла `*.list в /etc/apt/sources.list.d/`) и используя утилиту `apt`. Рассмотрим два последних варианта.
Начнём с ручного добавления репозитория. В текстовом редакторе открываем файл `/etc/apt/sources.list` и в конце файла вводим строку вида:

```
deb http://адрес_репозитория версия_дистрибутива ветки
```

Например, добавим репозиторий nginx, для этого создадим в каталоге `/etc/apt/source.list.d/` файл `nginx.list` следующего содержания:

```
deb http://nginx.org/packages/ubuntu jammy nginx
```

Здесь `jammy` — это версия Ubuntu, а nginx — название ветки, содержащей пакеты, необходимые для установки nginx. Следующий шаг — это установка публичного ключа репозитория, для этого нужна команда `apt-key`. Скачиваем при помощи `curl` наш ключ и передаём через `pipe` утилите `apt-key`:

```
curl -fsSL https://nginx.org/keys/nginx_signing.key | \
sudo apt-key add -
```

И последний шаг — это обновление информации о пакетах `sudo apt update` и установка пакета `sudo apt install nginx -y`.
Намного быстрее выглядит процесс добавления репозитория с помощью команды `apt-add-repository`. Эта команда автоматически создаёт записи в файле /etc/apt/sources.list или создаёт файл репозитория в каталоге `/etc/apt/sources.list.d/`, а также может удалять информацию о репозиториях. Чаще всего эта утилита используется для добавления PPA-репозиториев.
PPA-репозитории находятся на сайте Launchpad.net, который поддерживается компанией Canonical. Утилита автоматически находит строку для записи в файл репозитория, скачивает и импортирует ключи. Рассмотрим добавление репозитория nginx с использованием PPA-репозитория:

```
apt-add-repository ppa:nginx/stable
```

Здесь утилите `apt-add-repository` мы говорим, что подключаем PPA-репозиторий, поддерживаемый группой `nginx`, и подключаем стабильную версию. Утилита автоматически создаст файл

`/etc/apt/sources.list.d/nginx-ubuntu-stable-jammy.list` с содержимым, которое мы можем просмотреть при помощи команды cat. Утилита импортирует ключи и обновит список пакетов.

Поиск и установка deb-пакетов

Найти нужный пакет проще всего поиском в Интернете. Также пакет можно найти в консоли с помощью `apt search` или `apt list` (подробнее ниже). Пакет может находиться в стандартных репозиториях, тогда установить его легко (`apt install`). Если пакета нет в стандартных репозиториях, стоит поискать репозиторий от разработчика софта. При этом нужно удостовериться в совместимости репозитория с нашей версией ОС. В случае, если у разработчика нет репозитория под нужную версию ОС, то можно поискать сторонние репозитории (или PPA). Но при использовании сторонних репозиториев нужно учитывать риски доверия неофициальным источникам ПО.

Утилиты apt и dpkg

В Ubuntu управление пакетами осуществляется тремя способами: с использованием утилиты `apt`, `dpkg` или `snap`.

Самый удобный в использовании – **`apt`**. Это пакетный менеджер, который включает в себя набор утилит для управления пакетами. Он позволяет осуществлять поиск, установку и удаление пакетов, обновлять операционную систему, подключать репозитории. Рассмотрим параметры утилиты apt для управления пакетами:

- `apt list package_name –` поиск пакета по имени;
- `apt search package_name` — поиск пакета по имени и названию;
- `apt show package_name` — посмотреть информацию о пакете;
- `apt install package_name -y` — установить пакет;
- `apt install package_name1 package_name2 -y` — установить два пакета;
- `apt remove package_name` — удалить пакет, при этом сохранятся файлы с настройками;
- `apt purge package_name` — полностью удалить пакет, включая конфигурационные файлы;
- `apt upgrade` — обновить все установленные пакеты;
- `apt update` — обновить информацию о пакетах в репозиториях, указанных в настройках.

`dpkg` — пакетный менеджер в Debian-подобных системах. Главное отличие от утилиты apt состоит в том, что dpkg работает только с локальными пакетами, он не умеет искать и устанавливать пакеты из репозиториев. Основные параметры утилиты `dpkg`:

- `dpkg -l` — просмотр списка пакетов;
- `dpkg -i package_name` — установить пакет или группу пакетов;
- `dpkg -r package_name` — удалить пакет или группу пакетов.

Альтернативная система snap-пакетов

snap — это пакет, который, помимо готовой сборки самого приложения, включает в себя все необходимые зависимости и может работать почти в любом дистрибутиве Linux. В какой-то степени можно считать, что пакеты, установленные при помощи `snap`, — альтернатива самостоятельной сборке пакета. Пакет, установленный через `snap`, содержит все необходимые зависимости и может работать в любом окружении Linux. Snap состоит из двух частей: демона `snapd` и клиента для управления пакетами `snap`. Установка snapd производится командой `apt install snapd -y`. Варианты использования команды `snap`:

- `snap search package_name` — поиск пакета;
- `snap install package_name` — установка пакета;
- `snap refresh package_name` — обновление пакета;
- `snap remove package_name` — удаление пакета;
- `snap list` — просмотр установленных пакетов.

Планировщик cron

В ОС Linux есть два типа планировщиков: для выполнения разовых задач (например, перезагрузки сервера ночью) используют планировщик `at`, для выполнения задач с определённой периодичностью используют планировщик `cron`. Если выполнение разовых задач требуется довольно редко, то регулярные задачи крайне важны для работы любой системы. Сразу же после установки в Linux уже есть настроенные регулярные задачи. Как правило, это несложные команды или скрипты, которые выполняются по расписанию.
`Cron` — программа-демон, предназначенная для выполнения заданий в определённое время или через определённые промежутки времени. Список заданий, которые будут выполняться автоматически в указанные моменты времени, содержится в файле `/etc/crontab`, в `/etc/cron.d/*` и файлах `/var/spool/cron/*`. Рассмотрим подробнее системный файл `/etc/crontab`.

Системные файлы конфигурации crontab

В файле `/etc/crontab` находится общесистемная конфигурация задач по расписанию. Обычно, там определены задачи, которые должны выполняться от пользователя root. В начале файла можно определить переменные окружения, которые будут влиять на выполнение задач.
Далее указано расписание задач. Разберём пример задач, выполняемых каждый час:
`17 * * * * root cd / && run-parts --report /etc/cron.hourly`
Начинается запись с расписания (17-я минута каждого часа каждого дня). Далее указан пользователь (`root`) и после указана команда:
`cd / && run-parts --report /etc/cron.hourly`
Расписание имеет несколько параметров (слева направо):

1. минуты;
2. часы;
3. день месяца;

4. месяц;
5. день недели.

Минуты могут принимать значения от 0 до 59, часы — от 0 до 23, дни месяца — от 1 до 31, месяцы — от 1 до 12, дни недели — от 0 (воскресенье) до 6 (суббота). Дальше указываем пользователя (если делаем через утилиту `crontab`, это не нужно) и саму команду. Обратите внимание на `SHELL` и `PATH`. Не всё будет работать так же, как в консоли или скрипте.

Кроме числовых значений, доступны и другие знаки. Например, «`*`» определяет все допустимые значения. Если на месте всех значений звёздочки, скрипт будет запускаться каждую минуту, каждый день.

Через запятую (`,`) можно указать несколько значений: 1,3,4,7,8.

Тире (`-`) определяет диапазон значений, например, 1-6, что эквивалентно 1,2,3,4,5,6.

Звёздочка (`*`) определяет все допустимые значения поля. Например, звёздочка в поле «Часы» будет эквивалентна значению «каждый час».

Слеш (`/`) может использоваться для пропуска данного числа значений. Например, `*/3` в поле «Часы» эквивалентно строке `0,3,6,9,12,15,18,21`. Звёздочка означает «каждый час», но /3 диктует использовать только первое, четвёртое, седьмое (и так далее) значения, определённые звёздочкой. Например, каждые полчаса можно задать как `*/30`.

Минимальное время — одна минута. `Cron` каждую минуту просматривает список заданий и ищет те, которые нужно выполнить.

Отдельно стоит сказать о выводе команд. По умолчанию cron отправляет вывод скрипта на почту пользователя, который его запустил. Для любого локального пользователя можно настроить внешний ящик, куда будет отправляться предназначенная ему почта. Эти ящики можно вписать в конфиг `/etc/aliases`.

После его редактирования нужно запустить команду `newaliases`, настройку подробнее рассмотрим на последнем занятии. Поведение можно изменить, используя директиву `MAILTO`. Укажем имя пользователя или email, которому будет отправлено сообщение о выполнении задания:

`MAILTO=example@example.org`

Помимо системных задач в cron мы можем создавать пользовательские задачи. Основное отличие: их редактированием может заниматься сам пользователь и все задачи будут исполняться от его имени.

Пользовательские crontab

Для управления пользовательскими задачами следует использовать утилиту `crontab`. Как мы уже знаем, пользовательские файлы `crontab` нужны для настройки регулярных задач от обычных пользователей.

Посмотрим, как можно управлять пользовательскими задачами.

Вывести содержимое текущего файла расписания:

```
crontab -l
```

Удаление текущего файла расписания:

```
crontab -r
```

Редактирование текущего файла расписания. При первом запуске будет выведен список поддерживаемых текстовых редакторов:

```
crontab -e
```

Этот ключ позволяет выполнять вышеописанные действия для другого пользователя:

```
sudo crontab -u username
```

Файлы пользовательских crontab находятся в `/var/spool/cron`, там можно сразу найти всех пользователей с расписаниями (требуются `root`-права).

Итоги занятия

- Рассмотрели различные способы установки софта в ОС.
- Научились пользоваться основными пакетными менеджерами.
- Узнали, как можно найти пакет и подключить репозиторий.
- Изучили планировщик `cron` и настройку регулярных задач.