УЧЕБНОЕ ПОСОБИЕ

Системное администрирование РЕД ОС 7

Учебный центр Softline
Автор курса Лебедев Юрий

REDOS-102

softline®

Учебный центр

Автор: Лебедев Юрий
Контакты: yuriy.lebedev@softline.com

Название курса: Системное администрирование РЕД ОС 7
Учебное пособие

Код курса: REDOS-102

Версия: 1.0

Дата: Август, 2020 г.

Учебный центр Softline
http://softline.ru/edu

Условные обозначения

ПРИМЕЧАНИЕ

Описание каких-то нюансов или тонкостей, знание которых будет полезным.

ПРЕДУПРЕЖДЕНИЕ

Описание каких-то потенциально опасных особенностей, а также программ, неправильное применение которых может привести к сбоям или потерям данных.

ССЫЛКА НА ДОПОЛНИТЕЛЬНЫЙ РЕСУРС

Указание, где можно найти дополнительную полезную информацию по рассматриваемому вопросу.

РАСПРОСТРАНЕННАЯ ОШИБКА

Какой-то приём программирования, который часто вызывает проблемы, особенно у начинающих разработчиков.

Device > Setup > Operations

Шрифт используется для выделения элементов графического интерфейса в операционной системе РЕД ОС

```
uname -a
```

Шрифт используется для выделения команд интерфейса командной строки операционной системы РЕД ОС

О курсе

Курс «Системное администрирование РЕД ОС 7».

Продолжительность: 4 дня (32 а.ч.).

Модуль 1: Загрузка операционной системы РЕД ОС.

- Загрузчик GRUB2.
- Подсистема инициализации systemd.
- Управление службами.
- Управление целями загрузки.
- Лабораторная работа: Управление загрузкой.

Модуль 2: Диски, разделы и файловые системы РЕД ОС.

- Управление накопителями и разделами.
- Файловые системы ext и XFS.
- Проверка файловых систем.
- Управление пространством подкачки.
- Лабораторная работа: Управление дисками, разделами и файловыми системами.

Модуль 3: RAID и LVM в РЕД ОС.

- Типы логических накопителей.
- Программные RAID-массивы.
- Менеджер логических томов (LVM).
- Лабораторная работа: Управление RAID и LVM.

Модуль 4: Управление пользователями в РЕД ОС.

- Локальные пользователи и группы.
- Получение привилегий.
- Управление политиками паролей.
- Квоты на файловой системе.
- Лабораторная работа: Управление пользователями, привилегиями и квотами.

Модуль 5: Настройка сетевого взаимодействия в РЕД ОС.

- Управление сетевыми подключениями.
- Демон управления сетью NetworkManager.
- Настройка локального разрешения имен.
- Лабораторная работа: Управление сетью.

Модуль 6: Инструментальные средства системного администрирования РЕД ОС.

- Настройка сервера печати CUPS.
- Настройка и применение OpenSSH.
- Система ведения журналов Journald.
- Система ведения журналов Rsyslog.

- Лабораторная работа: Настройка удаленных сессий и журналов.

Модуль 7: Управление программным обеспечением РЕД ОС.

- Управление пакетами RPM.
- Управление репозиториями YUM.
- Сборка программ из исходных кодов.
- Лабораторная работа: Управление программным обеспечением.

Модуль 8: Ядро операционной системы РЕД ОС.

- Установка и обновление ядра.
- Управление параметрами ядра.
- Управление модулями ядра.
- Лабораторная работа: Управление ядром и модулями.

Модуль 9: Устранение неисправностей РЕД ОС.

- Подходы к устранению неисправностей.
- Устранение неисправностей загрузки системы.
- Устранение неисправностей накопителей и файловых систем.
- Устранение неисправностей сети.
- Лабораторная работа: Устранение неисправностей.

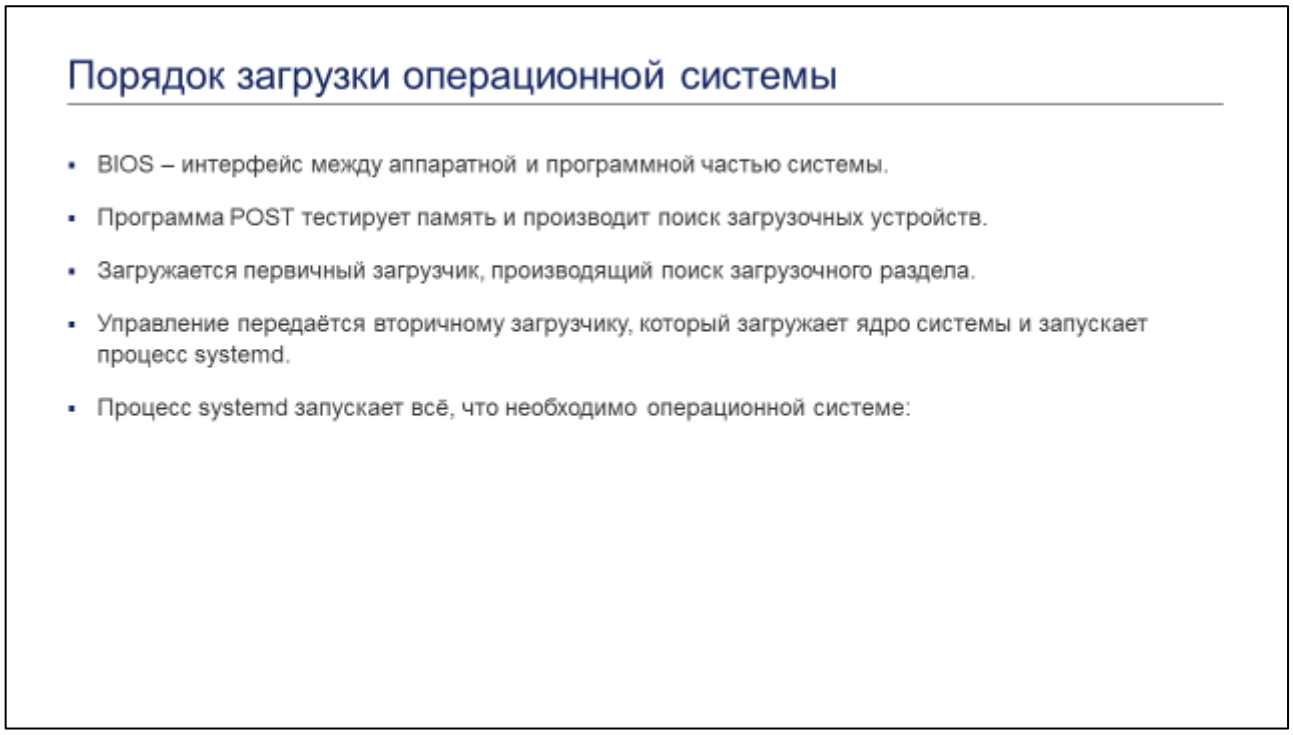
Модуль 10: Автоматизация установки РЕД ОС.

- Знакомство с возможностями Anaconda.
- Создание файла автоматических ответов.
- Автоматическая установка РЕД ОС.
- Лабораторная работа: Автоматическая установка РЕД ОС.

Модуль 1. Загрузка операционной системы РЕД ОС

В этом модуле рассматриваются:

- Загрузчик GRUB2
- Подсистема инициализации systemd
- Управление службами
- Управление целями загрузки



BIOS (Basic Input Output System) выполняет роль связующего между основными звеньями аппаратной вычислительной системы. BIOS предоставляет базовые инструкции нижнего уровня, используемые операционной системой.

При включении питания, BIOS запускает программу POST (Power On Self-Test), которая проверяет память, наличие различных устройств и так далее. После успешного завершения тестирования, программа POST выдаёт прерывание Int 19h и обработчик этого прерывания пытается найти загрузочные устройства. Загрузочных устройств может быть несколько, именно из них и будет выбираться устройство для загрузки, приоритеты могут быть заданы пользователям в настройках BIOS.

BIOS производит чтение первого сектора загрузочного устройства (как правило, это 512 байт), называемого Master Boot Record (MBR). MBR состоит из 3-х частей:

- 446 байт – первичный загрузчик.
- 64 байт – таблица разделов, хранящая 4 записи первичных разделов.
- 2 байта – магическое число, используемое для проверки того, что устройство является загрузочным.

Задача первичного загрузчика – найти в таблице разделов активный раздел и загрузить вторичный загрузчик, который находится на выбранном разделе, начиная с первого сектора.

Вторичный загрузчик осуществляет загрузку ядра и образа **`initrd`** (Initial Ram Disc), при необходимости.

Ядро, после загрузки, выполняет запуск программы systemd.

Загрузчик GRUB 2

- Конфигурационный файл:
  - **/boot/grub2/grub.cfg**
  - **/boot/efi/EFI/centos/grub.cfg**
- Утилита генерации **grub.cfg**:
  - **/usr/sbin/grub2-mkconfig**
  - Скрипты расположены в **/etc/grub.d/**.
  - Общие настройки расположены в **/etc/default/grub**.

РЕД ОС распространяется с GNU Grand Unified Boot Loader (GRUB) версии 2, который позволяет выбирать операционную систему или загрузчик в процессе загрузки системы. GRUB 2 также позволяет передавать аргументы ядру.

Настройка загрузчика GRUB 2.

GRUB2 читает настройки из файла **/boot/grub2/grub.cfg** или **/boot/efi/EFI/redhat/grub.cfg** на машинах с UEFI. Этот файл содержит информацию меню, однако, он не поддерживает редактирование, так как его генерирует утилита **/usr/sbin/grub2-mkconfig** в каталоге расположения ядра **/boot/**. Скрипты, расположены в **/etc/grub.d/**, а специальные настройки в файле **/etc/default/grub**. Любые ручные настройки будут потеряны в процессе обновления. Файл **grub.cfg** автоматически обновляется каждый раз, когда устанавливается новое ядро. Для обновления конфигурационного файла вручную, введите следующую команду с правами пользователя root:

```
# grub2-mkconfig -o /boot/grub2/grub2.cfg
```

Для систем с UEFI используйте:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Среди различных фрагментов кода и директив файл **grub.cfg** содержит один или более блоков **menuentry**. Каждый представляет собой одну запись в загрузочном меню GRUB 2. Эти блоки всегда начинаются с ключевого слова **menuentry**, далее название, список опций и блок в фигурных скобках. Все что находится между фигурными скобками должно быть написано с отступом. Следующий пример, демонстрирует блок **menuentry** для РЕД ОС 7.2 с ядром 4.19.79-1.el7.x86_64:

```
menuentry 'RED OS (4.19.79-1.el7.x86_64) MUROM (7.2)' --class red --class gnu-
linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-
4.19.79-1.el7.x86_64-advanced-768f657c-bb84-4d9b-9876-d7bbb78666d6' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
```

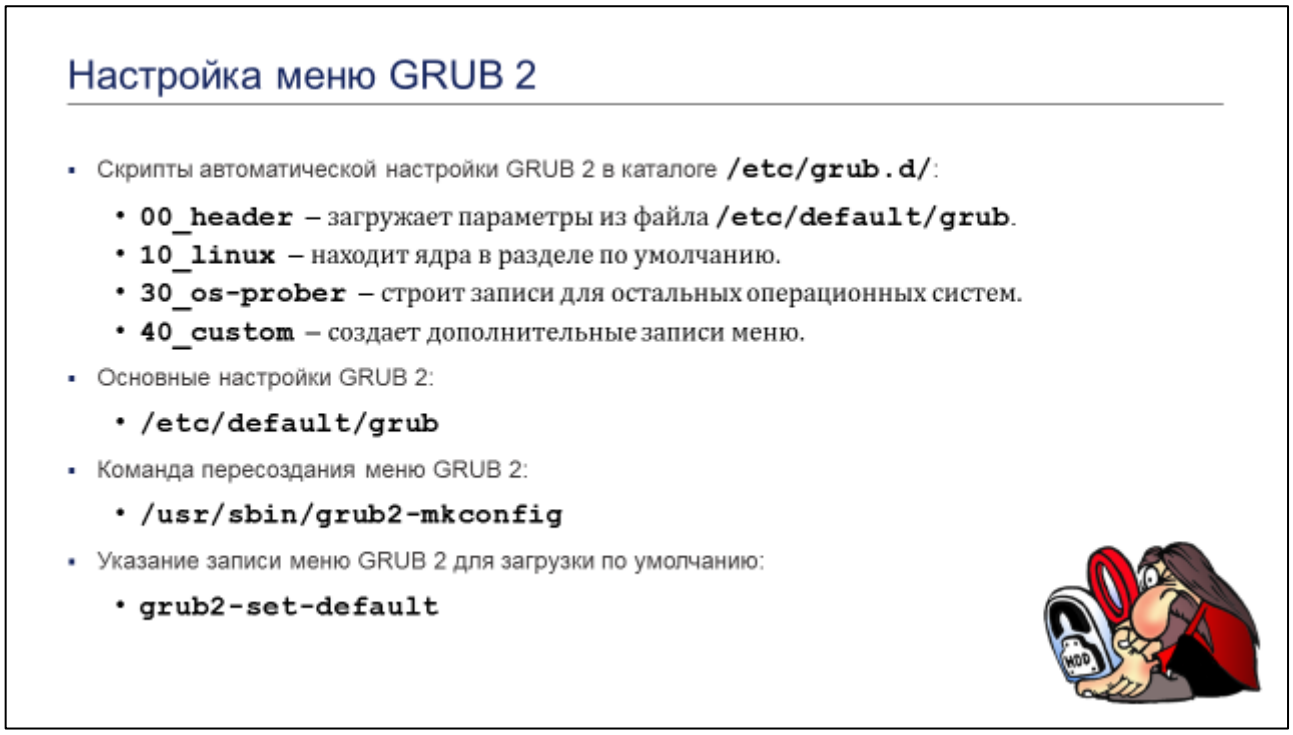
```
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-
ieee1275='ieee1275//sas/disk@0,msdos1' --hint-bios=hd0,msdos1 --hint-
efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  457ee40d-1fb0-
4c31-b6e6-a2917635fa59
        else
          search --no-floppy --fs-uuid --set=root 457ee40d-1fb0-4c31-b6e6-
a2917635fa59
        fi
        linux16 /vmlinuz-4.19.79-1.el7.x86_64 root=/dev/mapper/ro-root ro
rd.lvm.lv=ro/root rd.lvm.lv=ro/swap rhgb quiet LANG=ru_RU.UTF-8
        initrd16 /initramfs-4.19.79-1.el7.x86_64.img
}
```

Каждый блок **menuentry**, который представляет установленное ядро Linux содержит директивы **linux** (**linuxefi** на системах с EFI) и **initrd**, за ними следуют пути к ядру и образу **initramfs**. Если отдельный раздел **/boot** был создан, пути к ядру и образу **initramfs** записываются от **/boot**. В примере выше строка **initrd initrd16 /initramfs-4.19.79-1.el7.x86_64.img** обозначает, что образ **initramfs** на самом деле расположен в **/boot/initramfs-4.19.79-1.el7.x86_64.img**, когда корневая файловая система смонтирована, так же и с путем к ядру.

Версия ядра в строке **linux /vmlinuz-***ВЕРСИЯ* должна совпадать с версией образа **initramfs** в строке **initrd /initramfs-***ВЕРСИЯ***.img** каждого блока **menuentry**.

ПРИМЕЧАНИЕ

В блоке **menuentry** директива **initrd** должна указывать путь к файлу **initramfs** той же версии что и ядро. Директива называется **initrd**, потому что ранее использовались файлы **initrd**, создаваемые при помощи утилиты **mkinitrd**. Директива **initrd** осталась в файле **grub.cfg** для поддержки совместимости со старыми инструментами. Для создания начальных RAM-дисков теперь используется утилита **dracut**.



Настройка меню GRUB 2.

Скрипты GRUB 2 ищут на машине операционные системы и строят загрузочное меню. Для отражения новейших загрузочных опций, загрузочное меню автоматически перестраивается при каждом обновлении ядра или, когда новое ядро добавляется.

Однако, пользователи могут захотеть перестроить меню таким образом, чтобы добавить в него необходимые записи или изменить порядок записей. GRUB 2 позволяет базовую настройку загрузочного меню для предоставления управления отображаемым на экране.

GRUB 2 использует серию скриптов для построения меню. Они расположены в каталоге **`/etc/grub.d/`** и включают:

- **`00_header`**, который загружает параметры GRUB 2 из файла **`/etc/default/grub`**.
- **`10_linux`**, который находит ядра в разделе по умолчанию.
- **`30_os-prober`**, который строит записи для операционных систем, найденных на других разделах.
- **`40_custom`**, шаблон, который используется для создания дополнительных записей меню.

Скрипты из каталога **`/etc/grub.d/`** читаются в алфавитном порядке и могут быть переименованы для изменения порядка записей загрузочного меню.

ПРИМЕЧАНИЕ

Если вы установите ключ **`GRUB_TIMEOUT`** в значение **`0`** в файле **`/etc/default/grub`**, GRUB 2 не будет отображать список возможных для загрузки ядер во время старта системы. Для отображения списка загрузки зажмите цифровую или символьную клавишу сразу после отображения информации о BIOS и тогда GRUB 2 отобразит меню.

Смена загрузочной записи по умолчанию.

По умолчанию используется значение **saved** для ключа **GRUB_DEFAULT** в файле **/etc/default/grub**. Это инструктирует GRUB 2 загружать последнюю успешно загруженную операционную систему. Однако GRUB 2 не поддерживает числовые значения для этого ключа, для смены порядка по умолчанию, в котором операционные системы загружаются. Для указания, какая операционная систем должна быть загружена первой, используйте команду grub2-set-default, например:

```
# grub2-set-default 2
```

Обратите внимание, что позиции меню начинаются с **0**, таким образом предыдущая команда установила загрузку третей записи по умолчанию.

Перед перезагрузкой системы обновите конфигурационный файл, запустив команду:

```
# grub2-mkconfig –o /boot/grub2/grub.cfg
```

Или для систем с UEFI:

```
# grub2-mkconfig –o /boot/efi/EFI/redhat/grub.cfg
```

Редактирование записей.

Параметры ядра.

Для использования параметров ядра только в процессе загрузки, когда загрузочное меню GRUB 2 отображается перейдите к ядру, которые вы хотите запустить, нажмите клавишу e и отредактируйте строку с ядром, добавив в нее параметры ядра. Например, для запуска системы в спасательном режиме (Emergency Mode) добавьте параметр **emergency** в конец строки **linux**.

```
linux16 /vmlinuz-3.10.0-229.el7.x86_64 root=/dev/mapper/centos-root ro
rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet LANG=en_US.UTF-8 emergency
```

Однако, этот параметр применится только один раз, для того чтобы параметр остался, отредактируйте значения в ключе **GRUB_CMDLINE_LINUX** в файле **/etc/default/grub**. Например, вы можете включить аварийный режим (Emergency Mode) при каждой загрузке, отредактировав файл следующим образом:

```
GRUB_CMDLINE_LINUX=”emergency”
```

Обратите внимание, что вы можете указать несколько параметров для ключа **GRUB_CMDLINE_LINUX** также, как и при добавлении параметров, а загрузочном меню GRUB 2.

Затем запустите команду:

```
# grub2-mkconfig –o /boot/grub2/grub.cfg
```

Или для систем с UEFI:

```
# grub2-mkconfig –o /boot/efi/EFI/redhat/grub.cfg
```

Добавление новой записи.

Когда выполняется команда **grub2-mkconfig**, GRUB 2 ищет ядра Linux и другие операционные системы на базе файлов в каталоге **/etc/grub.d/**. Скрипт **10_linux** ищет установленные linux

ядра на том же разделе. **30_os-prober** ищет остальные операционные системы. Записи меню также автоматически добавляются в загрузочное меню при обновлении ядра.

Файл **40_custom** расположенный в каталоге **/etc/grub.d/** – это шаблон для записей создаваемых вручную и выглядит он следующим образом.

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```

Этот файл может быть отредактирован или скопирован. Запомните, что минимальная, корректная запись должна включать как минимум следующее:

```
Menuentry "ЗАГОЛОВОК"{
ДАННЫЕ
}
```

Использование только настроенных меню.

Если вы хотите использовать только настроенные записи меню, вы можете создать настраиваемое меню.

ПРЕДУПРЕЖДЕНИЕ

Перед началом данной процедуры сделайте резервную копию каталога **/etc/grub.d/** на случай, если придется вернуться назад позже.

ПРИМЕЧАНИЕ

Редактирование файла **/etc/default/grub** не оказывает эффекта при создании настраиваемого меню.

1. Скопируйте и вставьте содержимое файла **/boot/grub2/grub.cfg** или **/boot/efi/EFI/redhat/grub.cfg** в файл **/etc/grub.d/40_custom** после заголовка, так чтобы исполняемая часть скрипта **40_custom** осталась.
2. Удалите строки выше первой загрузочной записи до исполняемой части скрипта и ниже последней записи.
3. Удалите все файлы из каталога **/etc/grub.d**, кроме **00_header**, **40_custom** и **README**.

Вместо удаления файлов можно снять право выполнения на файлах при помощи команды **chmod a-x** *имя_файла*.

4. Отредактируйте, добавьте или удалите записи в **40_custom**.
5. Обновите файл **grub.cfg**, выполнив команду:

```
# grub2-mkconfig –o /boot/grub2/grub.cfg
```

Или для систем с UEFI:

```
# grub2-mkconfig –o /boot/efi/EFI/redhat/grub.cfg
```

Парольная защита GRUB 2

- **Типы пользователей:**
  - Суперпользователь – редактирование записей загрузчика.
  - Обычный пользователь – использование записей загрузчика.
- **Типы паролей:**
  - Обычный пароль.
  - Зашифрованный пароль (PBKDF2).

GRUB 2 поддерживает защиту при помощи пароля, которая использует незашифрованные пароли в шаблонах файлов GRUB 2. Для включения данного функционала, укажите супер-пользователя, который может использовать защищенные записи и пароль. Прочим пользователям, также может быть разрешен доступ к этим записям. Для указания к какие записи меню должны быть защищены паролем, отредактируйте файл **`/etc/grub.d/00_header`**. В качестве альтернативы, если вы хотите сохранить настройки после обновлений GRUB 2, измените файл **`/etc/grub.d/40_custom`**.

Настройка пользователей и паролей

- Конфигурационный файл: **/etc/grub.d/00_header**.
- Создание суперпользователя:
  - **set superusers="user01"**
  - **password user01 user01password**
- Создание обычного пользователя:
  - **password user02 user02password**
- Для применения настроек необходимо пересоздать **grub.cfg**:
  - **grub2-mkconfig**

Настройка пользовательской и парольной защиты, определение записей меню:

1. Для указания супер-пользователя добавьте следующие строки в файл **/etc/grub.d/00_header**, где **john** это имя пользователя оформленного, как супер-пользователь а **johnpassword** – это его пароль:

```
cat << EOF
set superusers="john"
password john johnpassword
EOF
```

2. Чтобы разрешить остальным пользователям доступ к записям меню, добавьте следующие строки для каждого пользователя в конец файла **/etc/grub.d/00_header**.

```
cat << EOF
set superusers="john"
password john johnpassword
password jane janepassword
EOF
```

3. После указания пользователей и паролей, укажите записи, которые должны быть защищены. Если вы не укажите записи, то все записи будут защищены паролем по умолчанию.
4. Обновите конфигурационный файл **grub.cfg**, выполнив команду:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Или для систем с UEFI:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Сохранение настроек

- Конфигурационный файл: **/etc/grub.d/40_custom**
  - **set superusers="user01"**
  - **password user01 user01password**
  - **password user02 user02password**
  - **menuentry 'REDOS' {…}**
  - **menuentry 'REDOS Safe Mode' --users user02 {…}**
- Для применения настроек необходимо пересоздать **grub.cfg**:
  - **grub2-mkconfig**

В качестве альтернативы, если вы хотите сохранить изменения после будущих обновлений GRUB2, добавьте строки в файл **/etc/grub.d/40_custom**, как показано в примере:

```
set superusers="john"
password john johnspassword
password jane janespassword
m enuentry 'RedOS' {
set root=(hd0,m sdos1)
linux /vm linuz
}
m enuentry 'RedOS Emerge' --user jane {
set root=(hd0,m sdos2)
linux /vm linuz
}
```

В примере выше, супер-пользователь (**superuser**) **john** может загружать любую запись меню и использовать командную строку меню GRUB 2 в процессе загрузки. В данном случае john может получить доступ к обоим записям: **RedOS** и **RedOS Emerge**. Любой может загрузить **RedOS**. Пользователь **jane** может загрузить запись **RedOS Emerge**, так как она указана в конфигурации. Если вы не укажите записи меню, то парольная защита не будет работать.

После внесения изменений, необходимо обновить конфигурационный файл **grub.cfg**, выполнив команду:

```
# grub2-mkconfig –o /boot/grub2/grub.cfg
```

Или для систем с UEFI:

```
# grub2-mkconfig –o /boot/efi/EFI/redhat/grub.cfg
```

Если вы не указали пользователя и пароль для записей меню, пароль супер-пользователя (**superuser**) будет запрашиваться при попытке доступа к системе.

Шифрование пароля

- Утилита генерации зашифрованного пароля:
  - **grub2-mkpasswd-pbkdf2**
- Конфигурационные файлы:
  - **/etc/grub.d/00_header**
  - **/etc/grub.d/40_custom**
    - **setsuperusers="john"**
    - **password_pbkdf2 john grub.pbkdf2.sha512.10000.19074739E…**

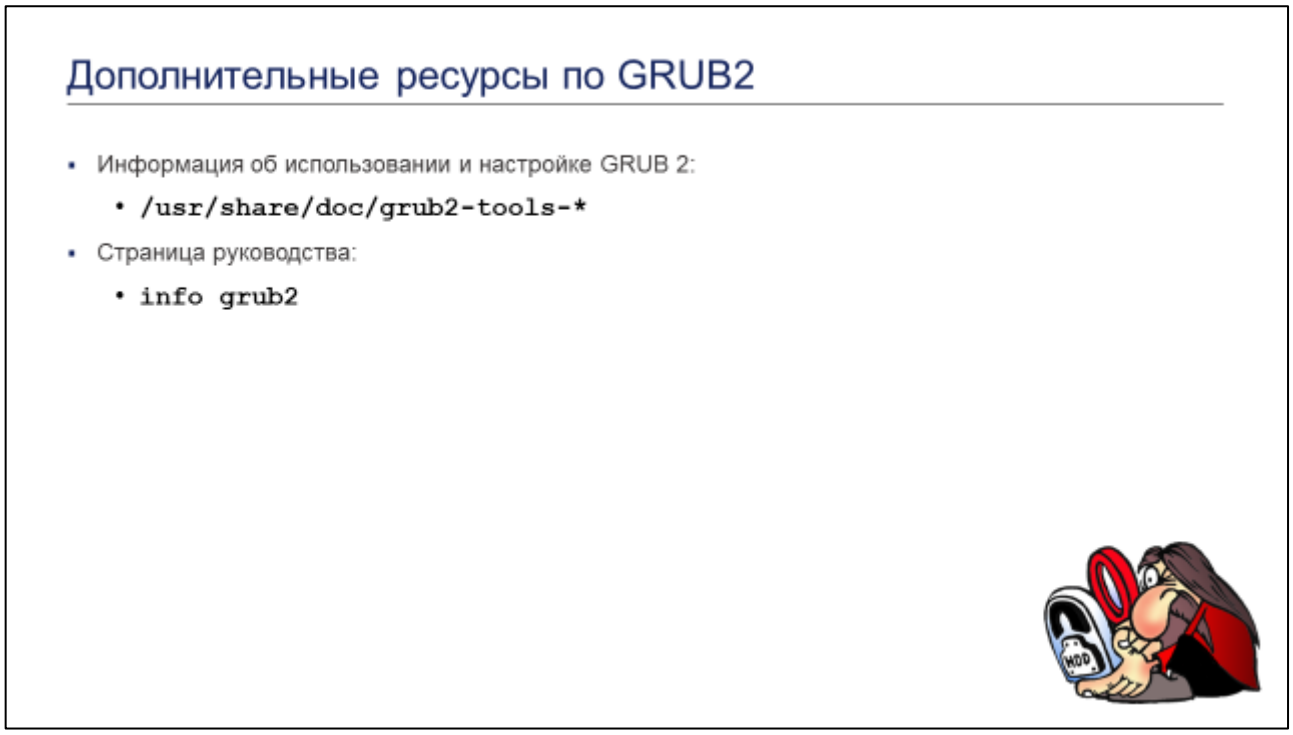
По умолчанию пароли сохраняются в виде плоского текста в скриптах GRUB 2. Нужно учесть, что к файлам нету доступа в процессе загрузки без указания корректного пароля, но безопасность может быть усилена за счет шифрования пароля при помощи команды **grub2-mkpasswd-pbkdf2**. Эта команда конвертирует указанный пароль в длинный хэш, который затем помещается в скрипты GRUB 2, вместо обычного текстового пароля.

1. Для генерации зашифрованного пароля используйте команду **grub2-mkpasswd-pbkdf2** от имени пользователя root.
2. Введите нужный пароль и повторите его после соответствующего запроса. Команда выдаст вам пароль в зашифрованном виде.
3. Скопируйте хэш и вставьте его в файл шаблона, где вы настраивали пользователя: **/etc/grub.d/00_header** или **/etc/grub.d/40_custom**. Следующий формат применяется для файла **00_header**:

```
cat <<EOF
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B014D86
2DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D53247EC0722DDF15FC.C56EC07389
11AD86CEA55546139FEBC366A393DF9785A8F44D3E51BF09DB980BAFEF85281CBBC56778D8B19DC948
33EA8342F7D73E3A1AA30B205091F1015A85
EOF
```

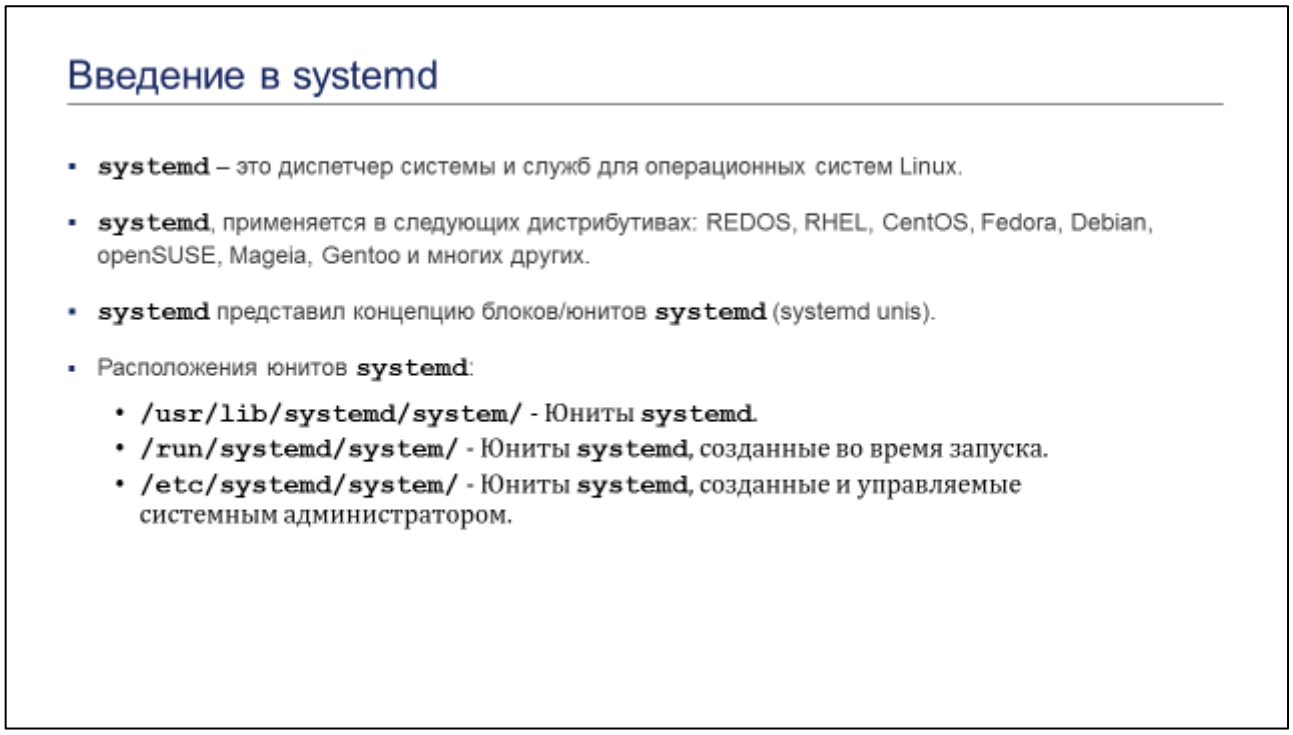
А следующий формат для файла **40_heder**:

```
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B014D86
2DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D53247EC0722DDF15FC.C56EC07389
11AD86CEA55546139FEBC366A393DF9785A8F44D3E51BF09DB980BAFEF85281CBBC56778D8B19DC948
33EA8342F7D73E3A1AA30B205091F1015A85
```



Для получения дополнительной информации по GRUB 2 воспользуйтесь следующими ресурсами:

- **/usr/share/doc/grub2-tools-***ВЕРСИЯ* - каталог содержит информацию об использовании и настройке GRUB 2.
- **info grub2** – Страница руководства, содержит общую информацию, руководства для пользователя и программиста, а также наиболее часто задаваемые вопросы и многое другое.



systemd – это диспетчер системы и служб для операционных систем Linux. Он разработан с возможностью обратной совместимости с инициализационными скриптами SysV и предоставляет набор возможностей, таких как параллельный запуск системных служб во время загрузки, запуск демонов по требованию, поддержка моментальных снимков состояния системы и логику управления службой, основанную на зависимостях. В RedOS 7, systemd используется как система инициализации по умолчанию.

systemd представил концепцию блоков/юнитов systemd (Systemd Units). Эти юниты представлены конфигурационными файлами юнитов, расположенных в одном из специальных каталогов (таблица расположения юнитов **systemd**, ниже), и инкапсулирующие информацию о системных службах, прослушиваемых сокетах, сохраненных моментальных снимках состояния системы и прочих объектах, которые имеют отношение к системе инициализации. Полный список юнитов systemd представлен в таблице ниже.

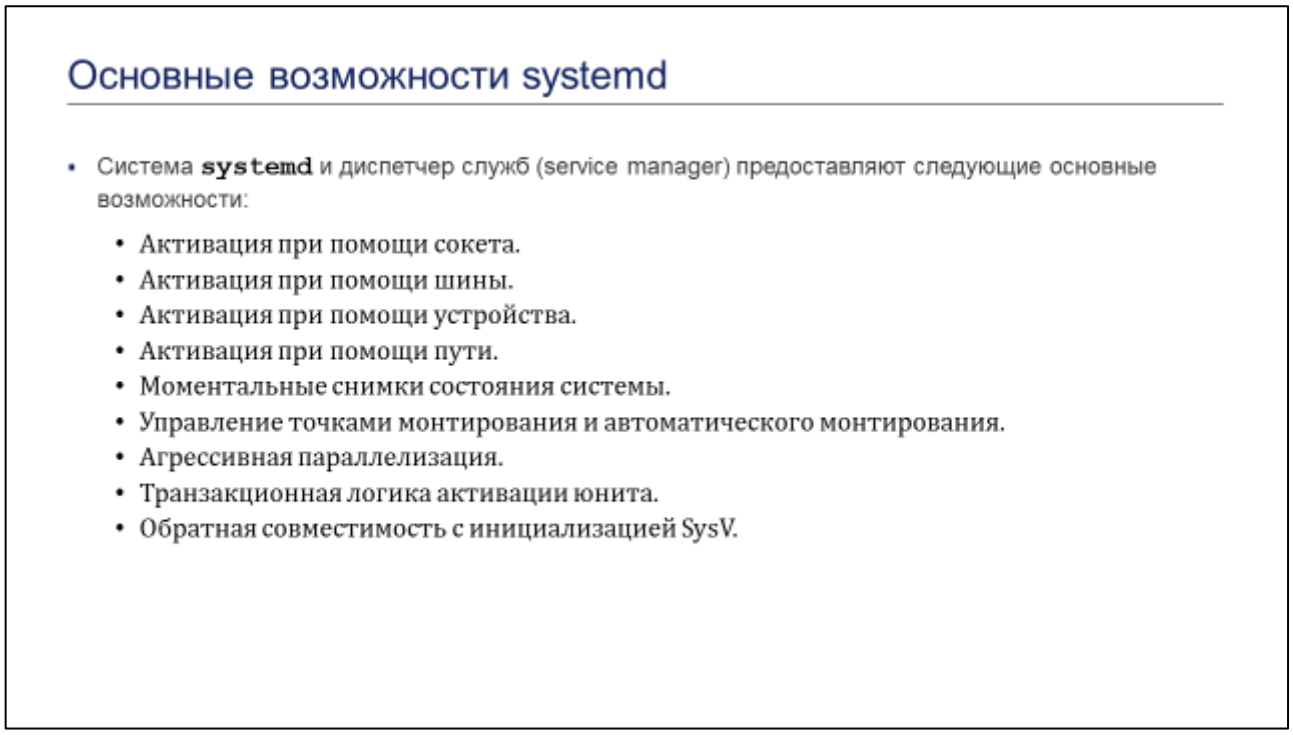
Доступные типы юнитов systemd (Systemd Units):

Тип юнита (Unit Type)	Расширение файла (File Extension)	Описание
Service unit	`.service`	Системная служба.
Target unit	`.target`	Группа системных служб.
Automount unit	`.automount`	Точка автоматического монтирования файловой системы.
Device unit	`.device`	Файл устройств, распознаваемых ядром.
Mount unit	`.mount`	Точка монтирования файловой системы.
Path unit	`.path`	Файл или каталог на файловой системе.
Scope unit	`.scope`	Процесс, созданный из вне.
Slice unit	`.slice`	Группа юнитов, организованные в иерархию, которые управляют системными процессами.
Snapshot unit	`.snapshot`	Сохраненное состояние диспетчера systemd.
Socket unit	`.socket`	Сокет коммуникаций внутри процесса.
Swap unit	`.swap`	Устройство или файл подкачки.

Timer unit	**.timer**	Таймер systemd.

Расположения юнитов **systemd**:

Каталог	Описание
/usr/lib/systemd/system/	Юниты **systemd**, распространяемые с устанавливаемыми RPM-пакетами.
/run/systemd/system/	Юниты **systemd**, созданные во время запуска. Этот каталог имеет приоритет перед каталогом с юнитами установленных служб.
/etc/systemd/system/	Юниты **systemd**, созданные и управляемые системным администратором. Этот каталог имеет приоритет перед каталогом с юнитами, созданными во время запуска.



В RedOS 7 система **systemd** и диспетчер служб (Service Manager) предоставляют следующие основные возможности:

Socket-based activation (Активация при помощи сокета) – В процессе загрузки, systemd создает прослушиваемый сокет для всех системных служб, которые поддерживают этот тип активации, и передает этот сокет этим службам, как только они запустятся. Это не только позволяет sysytemd запускать службы параллельно, но также делает возможным перезапуск службы без потери сообщений, отправленных службе в процессе перезапуска (пока она недоступна): соответствующий сокет остается доступным и принимает в очередь все сообщения.

Systemd использует юниты сокета (Socket units) для активации при помощи сокета.

Bus-based activation (Активация при помощи шины) – Системные службы используют D-Bus для коммуникации внутри процесса, могут быть запущены по требованию, когда приложение-клиент впервые обращается для коммуникации с ним.

Systemd использует файл службы D-Bus для активации при помощи шины.

Device-based activation (Активация при помощи устройства) – Системные службы, которые поддерживают активацию при помощи устройства, могут быть запущены по требованию, когда устройство конкретного типа подключено и становится доступно.

Systemd используют юниты устройств для активации при помощи устройства.

Path-based activation (Активация при помощи пути) – Системные службы, которые поддерживают активацию при помощи пути, могут быть запущены, когда конкретный файл или каталог изменяет свое состояние.

Systemd используют юниты пути для активации при помощи пути.

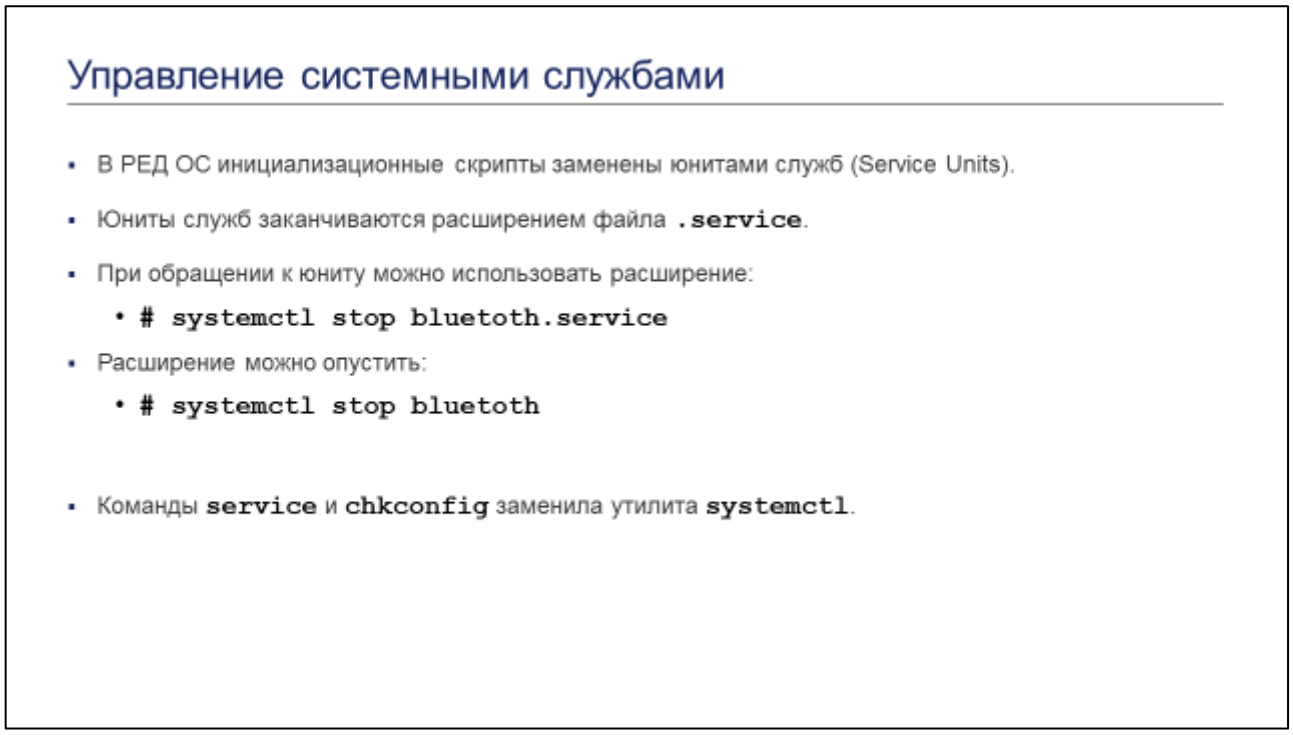
System state snapshots (Моментальные снимки состояния системы) – Systemd может временно сохранить текущее состояние всех юнитов или восстановить предыдущее состояние системы из динамически созданного моментального снимка юнитов.

Mount and automount point management (Управление точками монтирования и автоматического монтирования) – Systemd отслеживает и управляет точками монтирования и автоматического монтирования.

Aggressive parallelization (Агрессивная параллелизация) – За счет использования активации при помощи сокета, systemd может запускать системные службы параллельно, как только все прослушиваемые сокеты на месте. В комбинации с системными службами, которые поддерживают активацию по требованию, параллельная активация значительно сокращает время, необходимое для загрузки системы.

Transactional unit activation logic (Транзакционная логика активации юнита) – Прежде чем активировать или деактивировать юнит, systemd вычисляет его зависимости, создает временную транзакцию и проверяет ее консистенцию (целостность) транзакции. Если консистенция транзакции нарушена, systemd автоматически пробует скорректировать ее и удалить несущественные задания из нее, прежде чем сообщать об ошибке.

Backwards compatibility with SysV init (Обратная совместимость с инициализацией SysV) – Systemd полностью поддерживает скрипты инициализации SysV, как описано в Linux Standard Base Core Specification, что упрощает возможность обновления до юнитов служб systemd.



Раньше Linux распространялся с SysV или Upstart, которые используют инициализационные скрипты (Init Sctipts), расположенные в каталоге **/etc/rc.d/init.d/**. Обычно эти инициализационные скрипты написаны на Bash и позволяют системному администратору контролировать состояние служб и демонов в его системе. В RedOS 7 инициализационные скрипты заменены юнитами служб (Service Units).

Юниты служб заканчиваются расширением файла **.service** и служат той же цели, что и скрипты инициализации. Для просмотра, запуска, остановки, перезапуска, включения и отключения системных служб используйте команду systemctl как описано в таблице ниже. Команды **service** и **chkconfig** остались доступны в системе и работают корректно, но это сделано только для совместимости и рекомендуется отказаться от их использования.

ПРИМЕЧАНИЕ

Для ясности, все примеры далее используют полные имена юнитов с расширением **.service**, например:

```
# systemctl stop bluetoth.service
```

При работе с системными службами, можно пропустить расширение файла, для сокращения ввода. Когда утилита **systemctl** встречает имя юнита без расширения файла, она автоматически принимает его как юнит службы (Service Unit). Следующая команда равнозначна, приведенной выше:

```
# systemctl stop bluetoth
```

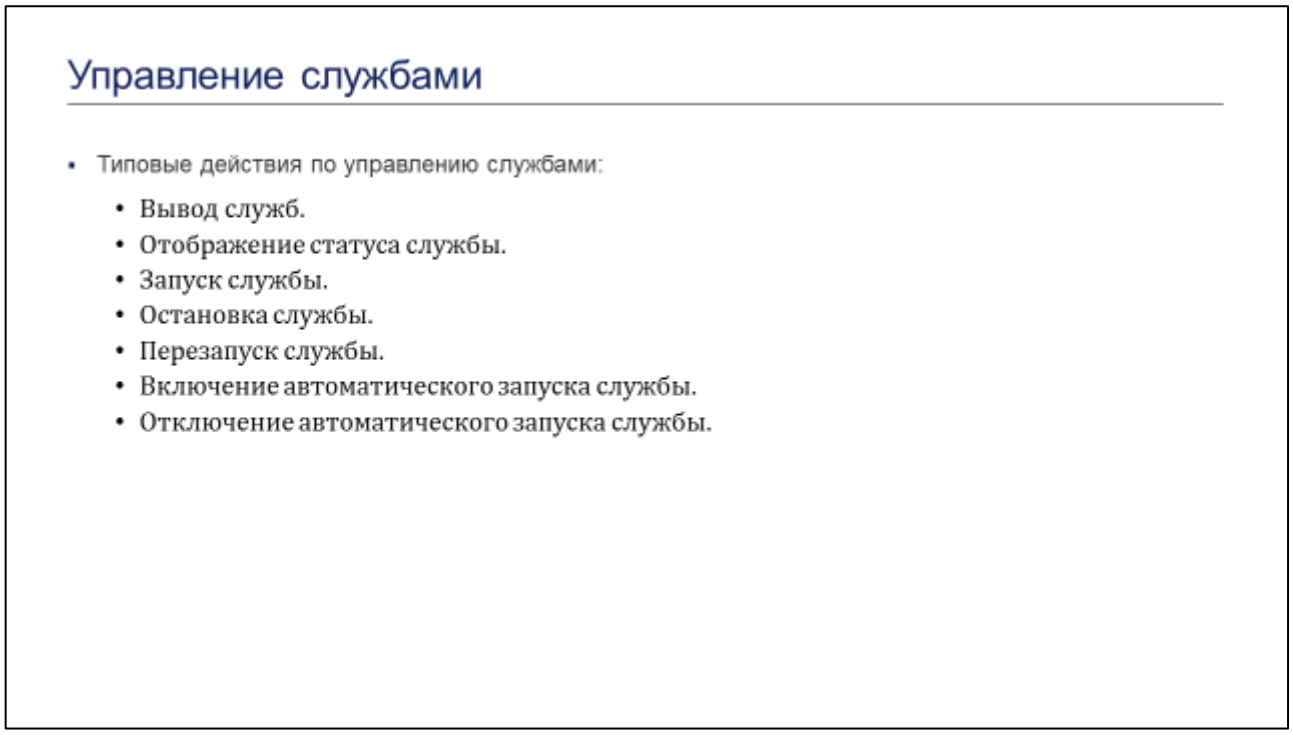
Таблица сравнения утилит **service** и **systemctl**:

service	systemctl
Запуск службы.	
service name start	systemctl start name.service

Остановка службы.	
service name stop	**systemctl stop name.service**
Перезапуск службы.	
service name restart	**systemctl restart name.service**
Перезапуск только запущенной службы.	
service name condrestart	**systemctl try-restart name.service**
Перезагрузка конфигурации	
service name reload	**systemctl reload name.service**
Проверка состояния службы.	
service name status	**systemctl status name.service** **systemctl is-active name.service**
Отображение статуса всех служб.	
service --status-all	**systemctl list-units --type service --all**

Таблица сравнения утилит **chkconfig** и **systemctl**:

chkconfig	**systemctl**
Включение автоматического запуска службы.	
chkconfig name on	**systemctl enable name.service**
Отключение автоматического запуска службы	
chkconfig name off	**systemctl disable name.service**
Проверка автоматического запуска службы.	
chkconfig --list name	**systemctl status name.service**
Вывод списка все служб с состоянием автоматического запуска.	
chkconfig –list	**systemctl list-unit-files --type service**



Вывод служб.

Для вывода всех загруженных юнитов служб, введите следующую команду:

```
$ systemctl list-units –type service
```

Для каждого юнита службы, эта команда отобразит его полное имя (**UNIT**), далее следует заметка о состоянии загрузки юнита (**LOAD**), его состояние высокого уровня (high-level) (**ACTIVE**) и низкого уровня (low-level) (**SUB**) и короткое описание (**DESCRIPTION**).

По умолчанию команда **systemctl list-units** отображает только активные юниты. Если вы хотите увидеть список всех загруженных юнитов вне зависимости от их состояния, запустите команду с опцией **--all** или **-a**.

```
# systemctl list-units --type service --all
```

Также вы можете вывести список всех доступных юнитов служб, чтобы увидеть состояние их автозапуска. Используйте следующую команду:

```
# systemctl list-unit-files --type service
```

Для каждого юнита службы эта команда отобразит его полное имя (**UNIT FILE**), далее будет указано состояние автоматического запуска (**STATE**).

Отображение статуса службы.

Для отображения подробной информации о юните службы, который связан с системной службой, введите следующую команду:

```
$ systemclt status ИМЯ.service
```

Замените name на имя юнита службы информацию о которой хотите исследовать (например, **gdm**). Эта команда выведет имя выбранного служебного юнита и его короткое описание, а также одно

или более поле, описанных в таблице ниже. Если команда выполнена от имени пользователя root, также будут выведены последние записи журнала.

Доступная информация о юните службы.

Поле	Описание
Loaded	Информация о состоянии загрузки юнита (loaded), абсолютном пути к файлу юнита и отметка об автоматическом запуске службы.
Active	Информация о текущем состоянии запуска юнита службы с временной отметкой.
Main PID	PID соответствующей системной службы и ее имя.
Status	Дополнительная информация о системной службе.
Process	Дополнительная информация о связанном процессе.
CGroup	Дополнительная информация о связанной группе управления (Control Groups).

Чтобы определить только состояние запуска службы, используйте:

```
# systemctl is-active ИМЯ.service
```

Чтобы определить только состояние автозапуска службы, используйте:

```
# systemctl is-enabled ИМЯ.service
```

Запуск службы.

Для запуска юнита службы, который связан с системной службой, введите следующую команду:

```
# systemctl start ИМЯ.service
```

Замените name именем юнита службы, которую вы хотите запустить (например, **gdm**). Эта команда запускает выбранный юнит службы в текущей сессии.

Остановка службы.

Для остановки юнита службы, который связан с системной службой, введите следующую команду:

```
# systemctl stop ИМЯ.service
```

Замените name именем юнита службы, которую вы хотите остановить (например, bluetoth). Эта команда остановит выбранный юнит службы в текущей сессии.

Перезапуск службы.

Для перезапуска юнита службы, который связан с системной службой, введите следующую команду от имени пользователя root:

```
# systemctl restart ИМЯ.service
```

Замените name именем юнита службы, который хотите перезапустить (например, httpd). Эта команда остановит выбранный юнит службы в текущей сессии и немедленно запустите его снова. Нужно отметить, что, если текущий юнит службы не запущен, данная команда все равно его запустит. Для того чтобы команда сработала только в том случае, когда соответствующая служба запущена, выполните ее следующим образом от имени root:

```
# systemctl try-restart ИМЯ.service
```

Некоторые службы поддерживают перезагрузку конфигурационных файлов без воздействия на выполнение службы, выполните следующую команду от имени пользователя root:

```
# systemctl reload ИМЯ.service
```

Обратите внимание, что, если системна служба не поддерживает данную возможность, она проигнорирует всю команду целиком. Для удобства команда **systemctl** поддерживает команды **reload-or-restart** и **reload-or-try-restart,** которые перезапускают службы, не поддерживающие перезагрузку конфигурационных файлов.

Включение автоматического запуска службы.

Для настройки юнита службы на автоматический старт в процессе загрузки, выполните следующую команду от имени пользователя root:

```
# systemctl enable ИМЯ.service
```

Замените name именем юнита службы, который хотите настроить на автоматический запуск (например, **httpd**). Эта команда читает секцию **[Install]** выбранного юнита службы и создает необходимую символьную ссылку к файлу **/usr/lib/system/***ИМЯ***.service** в каталоге **/etc/system/system/** и его подкаталогах. Однако эта команда не переписывает ссылку если такая уже существует. Если вы хотите убедиться, что ссылка вновь создана, используйте следующую команду от имени root:

```
# systemctl reenable ИМЯ.service
```

Данная команда отключает выбранный юнит службы и вновь включает его снова.

Отключение автоматического запуска службы.

Для отключения автоматического запуска юнита службы и соответствующей системной службы, введите следующую команду от имени пользователя root:

```
# systemctl disable ИМЯ.service
```

Замените name именем юнита службы, который хотите отключить от автоматического запуска (например, **bluetooth**). Этак команда читает секцию **[Install]** выбранного юнита службы и удаляет необходимую символьную ссылку к файлу **/usr/lib/system/***ИМЯ***.service** из каталога **/etc/system/system/** и его подкаталогах. Дополнительно вы можете замаскировать любой юнит службы, чтобы предотвратить его запуск вручную иной службой. Для этого используйте следующую команду от имени root:

```
# systemctl mask ИМЯ.service
```

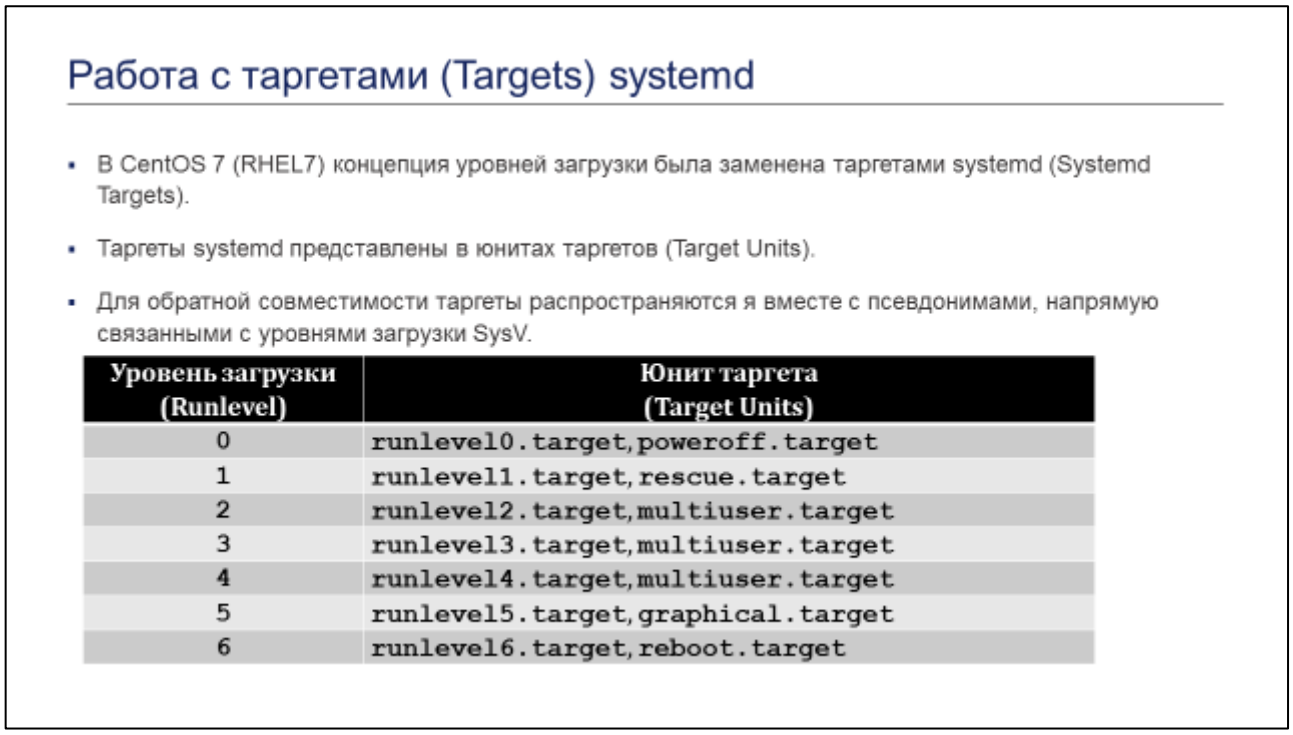
Данная команда заменит файл **/etc/system/system/***ИМЯ***.service** на символьную ссылку к **/dev/null**, делая оригинальный файл недоступным для systemd. Чтобы отменить это действие снять маскировку с юнита службы, используйте следующую команду от имени пользователя root:

```
# systemctl unmask ИМЯ.service
```

Пример отключения автоматического запуска службы.

Чтобы отключить автоматический запуск юнита службы в текущей сессии, выполните следующую команду, от имени пользователя root:

```
# systemctl disable ИМЯ.service
```



Более ранние версии Linux, которые распространялись с SysV init или Upstart, использовали определенный заранее набор уровней загрузки (**`runlevel`**), которые представляли собой специфические режимы выполнения. Эти уровни загрузки нумеровались от **0** до **6** и определялись системными службами, которые должны быть запущены на соответствующем уровне загрузки. В RedOS 7 концепция уровней загрузки была заменена таргетами systemd (Systemd Targets).

Таргеты systemd представлены в юнитах таргетов (Target Units). Юниты таргетов заканчиваются расширением файла **`.target`** и их основное назначение в группировке юнитов system в цепочку зависимостей. Например, юнит graphical.target, который используется для запуска графической сессии, запускает системные службы, таки как GNOME Display Managaer (**`gdm.service`**) или Account Service (**`account-daemon.service`**) и также активирует **`multi-user.target`**. Аналогично **`multi-user.target`** активирует прочие основные системные службы, такие как NetworkManager (**`NetworkManager.service`**) или D-Bus (**`dbus.service`**) и активирует еще один юнит таргета – **`basic.target`**.

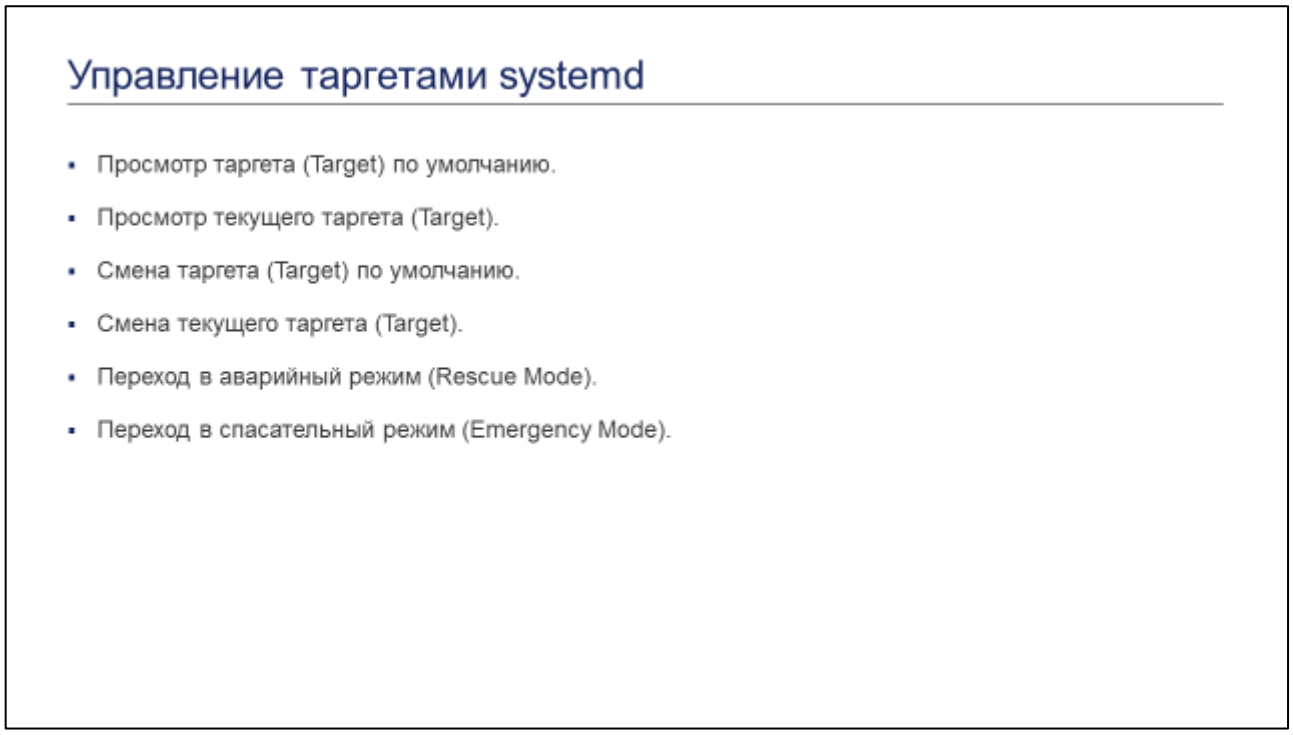
RedOS 7 распространяется с определенным заранее набором таргетов, которые отчасти похожи на стандартные уровни загрузки из предыдущего релиза системы. Для обратной совместимости таргеты распространяются я вместе с псевдонимами, напрямую связанными с уровнями загрузки SysV.

Уровень загрузки (Runlevel)	Целевые юниты (Target Units)	Описание
0	**`runlevel0.target, poweroff.target`**	Завершение работы и выключение системы.
1	**`runlevel1.target, rescue.target`**	Запуск аварийной оболочки.
2	**`runlevel2.target, multiuser.target`**	Запуск многопользовательской системы без графического окружения.
3	**`runlevel3.target, multiuser.target`**	Запуск многопользовательской системы без графического окружения.

4	**runlevel4.target, multiuser.target**	Запуск многопользовательской системы без графического окружения.
5	**runlevel5.target, graphical.target**	Запуск многопользовательской системы с графическим окружением
6	**runlevel6.target, reboot.target**	Завершение работы и перезагрузка системы.

Для просмотра, изменения или настройки таргетов systemd используйте утилиту **systemctl**. Команды **runlevel** и **telinit** остались доступны в системе только для совместимости, поэтому постарайтесь исключить их из работы.

Устаревшая команда	Новая команда	Описание
runlevel	**systemctl list-units --type target**	Список загруженных юнитов таргетов.
telinit *УРОВЕНЬ*	**systemctl isolate** *ИМЯ*.**target**	Смена текущего таргета.



Просмотр таргета (Target) по умолчанию.

Чтобы определить какой юнит таргета используется по умолчанию выполните следующую команду:

```
# systemctl get-default
```

Данная команда проверяет назначение символьной ссылки **`/etc/systemd/system/default.target`** и отображает результат.

Просмотр списка доступных таргетов (Targets).

Для вывода списка всех загруженных на данный момент юнитов таргетов, введите следующую команду:

```
# systemctl list-units --type target
```

Для каждого юнита таргета эта команда отображает его полное имя [**`UNIT`**], указывая примечание о том, что юнит загружен [**`LOAD`**], его верхний уровень (High-Level) [**`ACTIVE`**] и нижний уровень (Low-Level) [**`SUB`**] состояния активации юнита и короткое описание [**`DESCRIPTION`**]

По умолчанию команда **`systemctl list-units`** отображает только активные юниты. Если вы хотите вывести список всех загруженных юнитов вне зависимости от их состояния, выполните команду вместе с опицией **`--all`** или **`-a`**:

```
# systemctl list-units --type target --all
```

Смена таргета (Target) по умолчанию.

Для настройки системы на использование другого таргета по умолчанию, введите следующую команду от имени пользователя root:

```
# systemctl set-default ИМЯ.target
```

Замените *ИМЯ* именем юнита таргета, который вы хотите использовать по умолчанию (например, **multi-user**). Эта команда заменяет файл **/stc/systemd/system/default.target** символьной ссылкой к **/usr/lib/systemd/system/name.target**, где name это имя юнита таргета, который вы собираетесь использовать.

Смена текущего таргета (Target).

Для переключения на другой таргет в рамках текущей сессии, используйте следующую команду от имени пользователя root:

```
# systemctl isolate ИМЯ.target
```

Замените *ИМЯ* на имя юнита таргета, который вы хотите использовать (например, **multi-user**). Эта команда запускает юнит таргета name и все зависимые юниты, все остальные останавливает.

Переход в аварийный режим (Rescue Mode).

Аварийный режим (Rescue Mode) – предоставляет удобный однопользовательское окружение, которое позволит вам восстановить вашу систему в случае невозможности загрузиться в нормальное состояние. В аварийном режиме система пытается смонтировать все локальные файловые системы и запускает только важные системные сервисы, но не запускает сетевые интерфейсы и не позволяет параллельный вход в систему. В RedOS 7 аварийный режим полностью соответствует однопользовательскому режиму (Single User Mode) и требует ввода пароля пользователя root.

Чтобы сменить текущий таргет и войти в аварийный режим в текущей сессии, введите следующую команду от имени пользователя root:

```
# systemctl rescue
```

Эта команда похожа на **systemctl isolate rescue.target**, но она также посылает информационное сообщение всем пользователям, которые на данный момент находятся в системе. Чтобы предотвратить отправку сообщения, введите команду с указанием опции **--no-wall**:

```
# systemctl --no-well rescue
```

Переход в спасательный режим (Emergency Mode).

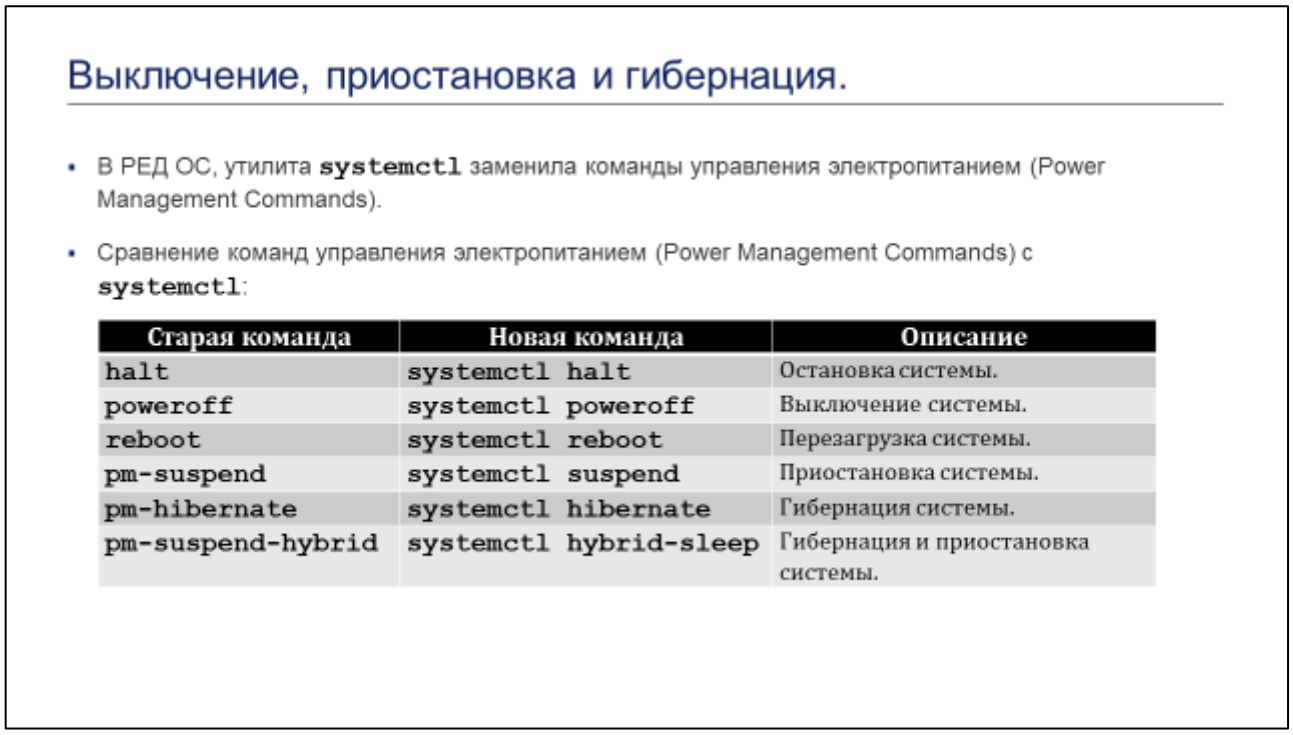
Спасательный режим (Emergency Mode) – предоставляет наиболее минимальное окружение из возможных и позволяет вам восстанавливать вашу систему даже в ситуации, когда система не может войти в аварийное окружение (Rescue Mode). В спасательном окружении система монтирует корневую файловую систему только для чтения, не пробует монтировать остальные локальные файловые системы, не активирует сетевые интерфейсы и запускает только несколько ключевых сервисов. В RedOS 7 спасательный режим требует пароль пользователя root.

Для смены текщего таргета и входа в спасательный режим, введите следующую команду от имени пользователя root:

```
# systemctl emergency
```

Эта команда похожа на **systemctl isolate emergency.target**, но она также посылает информационное сообщение всем пользователям, которые на данный момент находятся в системе. Чтобы предотвратить отправку сообщения, введите команду с указанием опции **--no-wall**:

```
# systemctl --no-well emergency
```



В RedOS 7, утилита systemctl заменила команды управления электропитанием (Power Management Commands) используемыми в более ранних версиях Linux. Оригинальные команды оставлены в составе системы для обеспечения совместимости, но постарайтесь от них отказаться в пользу **`systemctl`** если это возможно.

Сравнение команд управления электропитанием (Power Management Commands) с **`systemctl`**.

Старая команда	Новая команда	Описание
`halt`	`systemctl`	Остановка системы.
`poweroff`	`systemctl poweroff`	Выключение системы.
`reboot`	`systemctl reboot`	Перезагрузка системы.
`pm-suspend`	`systemctl suspend`	Приостановка системы.
`pm-hibernate`	`systemctl hibernate`	Гибернация системы.
`pm-suspend-hybrid`	`systemctl hybrid-sleep`	Гибернация и приостановка системы.

Выключение системы.

Для выключения системы введите следующую команду с правами пользователя root:

```
# systemctl poweroff
```

Для выключения и остановки системы без выключения машины введите следующую команду от имени пользователя root:

```
# systemctl halt
```

По умолчанию обе эти команды заставляют **`systemd`** отправить информационное сообщение всем пользователям, находящимся в системе. Чтобы предотвратить отправку сообщений, выполните команду с указанием опции **`--no-wall`**:

```
# systemctl --no-wall poweroff
```

Перезагрузка системы.

Для перезагрузки системы введите следующую команду с правами пользователя root:

```
# systemctl reboot
```

По умолчанию эта команды заставляют **systemd** отправить информационное сообщение всем пользователям, находящимся в системе. Чтобы предотвратить отправку сообщений, выполните команду с указанием опции **--no-wall**:

```
# systemctl --no-wall reboot
```

Приостановка системы.

Для приостановки системы, введите следующую команду от имен пользователя root:

```
# systemctl suspend
```

Эта команда сохраняет состояние системы в оперативной памяти и за исключением модуля оперативной памяти выключает устройства машины. Когда вы вновь включаете машину, система восстанавливает состояние без необходимости выполнять процесс загрузки. Потому как состояние системы сохранено в оперативной памяти, а не на жестком диске, восстановление системы из происходит значительно быстрее, чем при гибернации. С другой стороны, приостановка системы чувствительна к потри электропитания.

Гибернация системы.

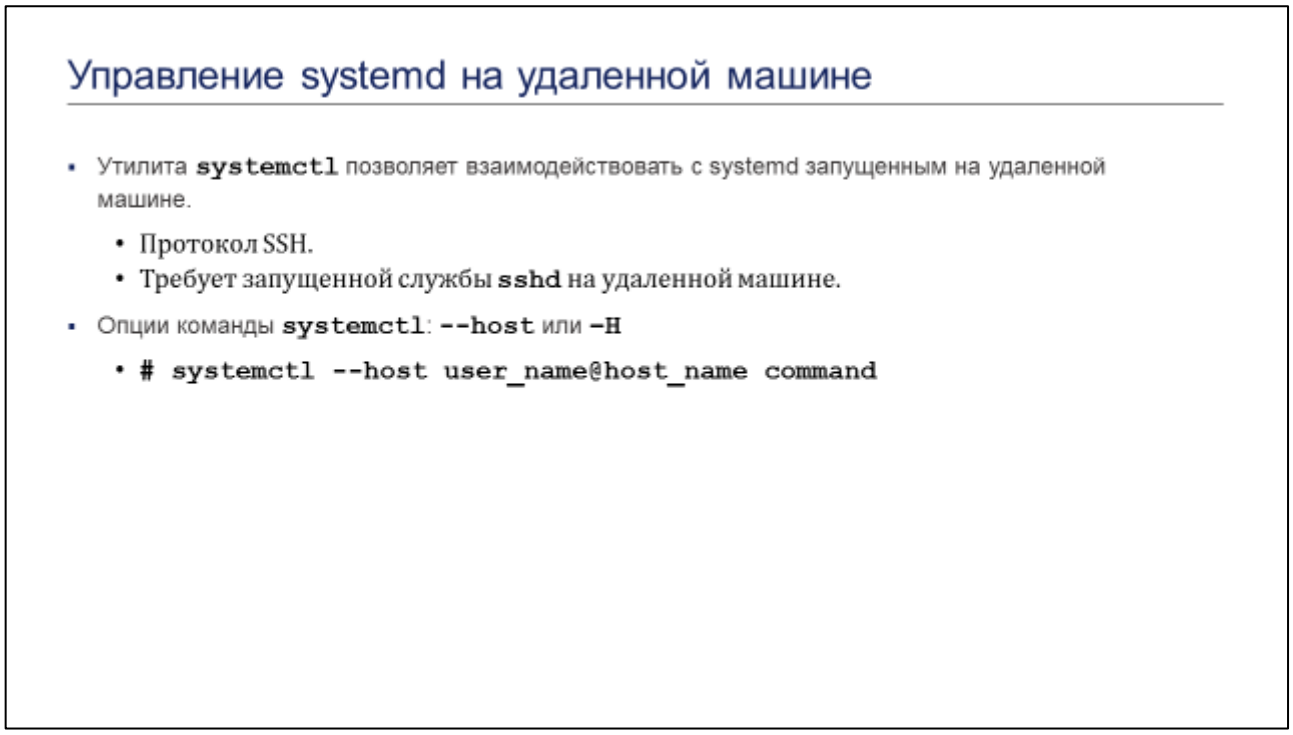
Для гибернации системы, введите следующую команду от имен пользователя root:

```
# systemctl hibernate
```

Эта команда сохраняет состояние системы на жестком диске и выключает машину. Когда вы вновь включаете машину, система восстанавливает состояние из данных, сохраненных на диске, без необходимости загрузки. Так как данные сохранены на жестком диске, машина не нуждается в энергоснабжении, но этот процесс занимает больше времени чем восстановление из состояния приостановки.

Для одновременной гибернации и приостановки системы введите следующую команду от имени пользователя root:

```
# systemctl hybrid-sleep
```



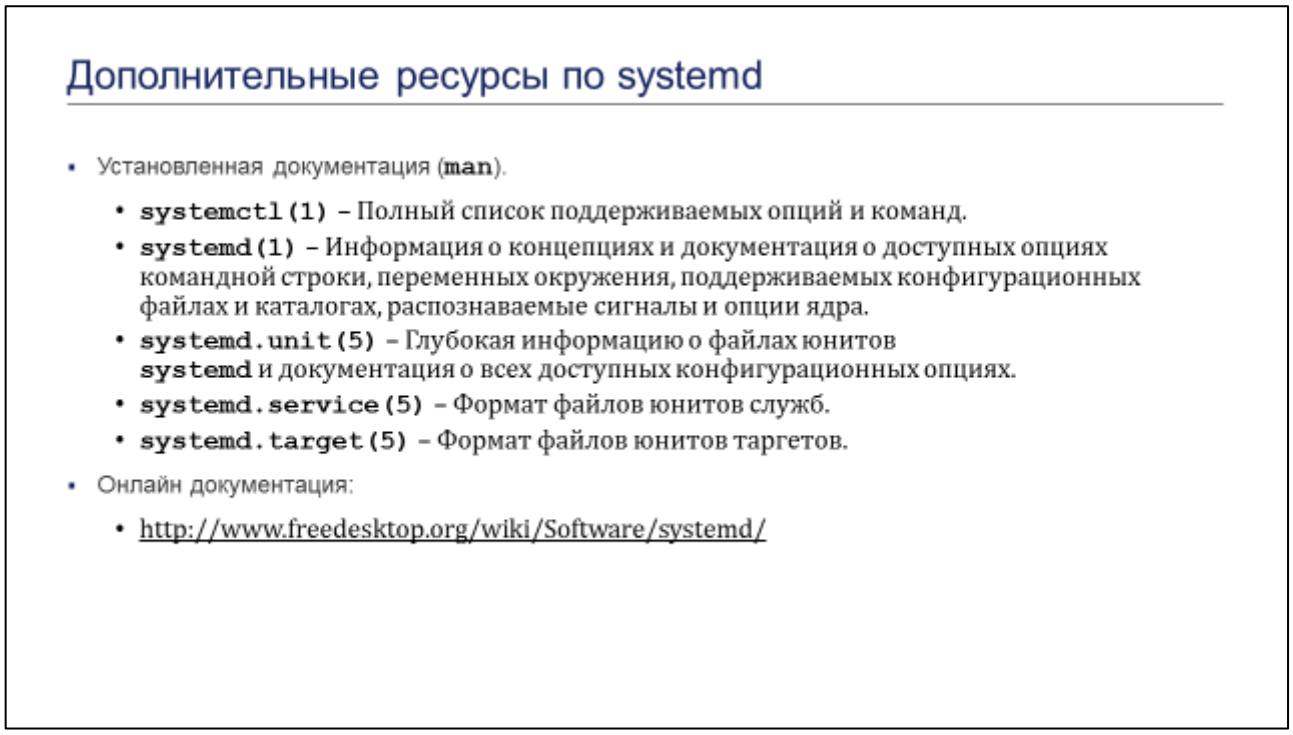
В дополнение к управлению системой systemd и службами локально, утилита systemctl также позволяет вам взаимодействовать с systemd запущенным на удаленной машине по протоколу SSH. При условии, что служба sshd на удаленной машине запущена, вы можете подключиться к этой машине при помощи команды systemctl с указанием опций --host или -H:

```
# systemctl --host ПОЛЬЗОВАТЕЛЬ@КОМПЬЮТЕР КОМАНДА
```

Замените *ПОЛЬЗОВАТЕЛЬ* именем удаленного пользователя, *КОМПЬЮТЕР* именем хоста и *КОМАНДА* любой командой **systemctl**, описанной выше.

ПРИМЕЧАНИЕ

Удаленная машина должна быть сконфигурирована для разрешения пользователю подключаться удаленно при помощи сервера SSH.



Для получения дополнительной информации о **systemnd** и его применении в RedOS 7, воспользуйтесь следующими ресурсами:

Установленная документация:

- **systemctl(1)** – Страница руководства для команды systemctl содержит полный список поддерживаемых опций и команд.
- **systemd(1)** – Страница руководства для системы systemd и диспетчера служб содержит больше информации о концепциях и документирует доступные опции командной строки, переменные окружения, поддерживаемые конфигурационные файлы и каталоги, распознаваемые сигналы и опции ядра.
- **systemd.unit(5)** – Страница руководства содержит глубокую информацию о файлах юнитов systemd и документирует все доступные конфигурационные опции.
- **systemd.service(5)** – Страница руководства содержит формат файлов юнитов служб (Service Units).
- **systemd.target(5)** – Страница руководства содержит формат файлов юнитов таргетов (Target Units).

Онлайн документация:

http://www.freedesktop.org/wiki/Software/systemd/ – домашняя страница проекта, содержит больше информации о **systemd**.

Модуль 2. Диски, разделы и файловые системы РЕД ОС

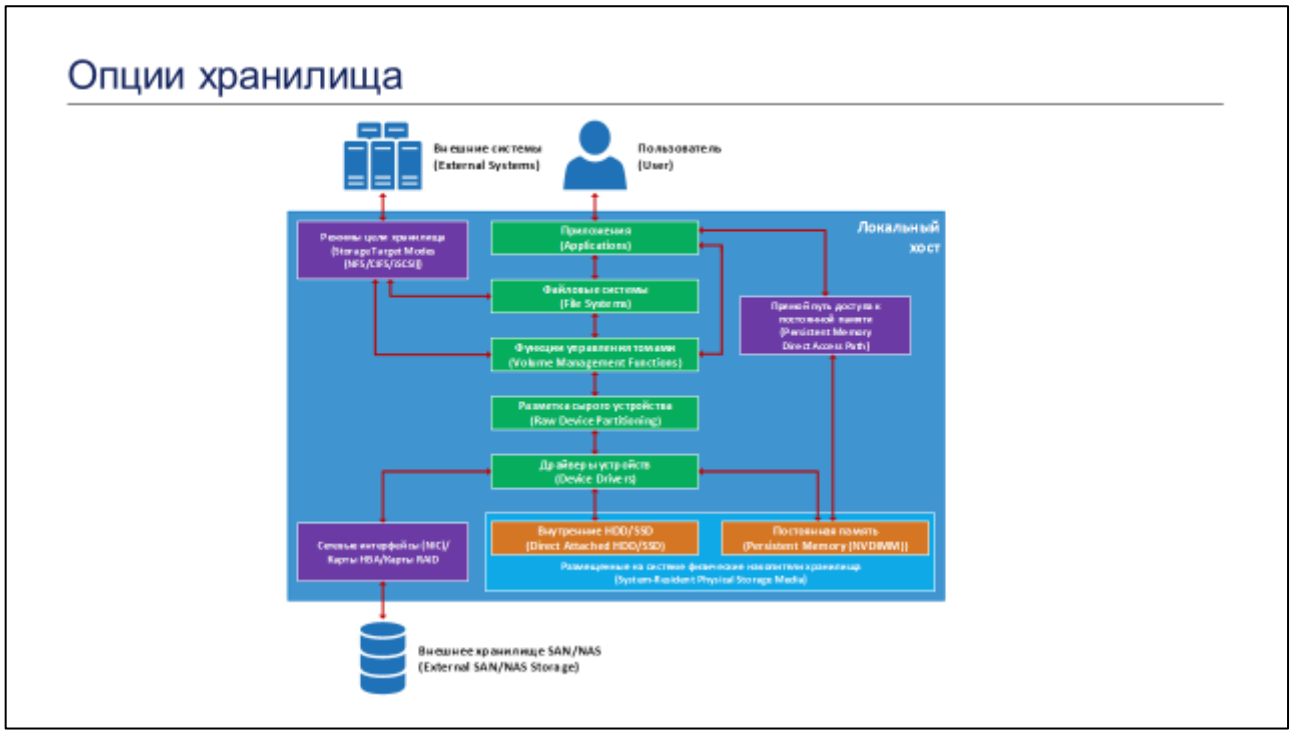
В этом модуле рассматриваются:

Управление накопителями и разделами

Файловые системы ext и XFS

Резервное копирование файловых систем

Проверка файловых систем



В RedOS 7 доступны следующие опции локального хранилища:

Администрирование базовых дисков:

- При помощи **`parted`** и **`fdisk`** можно создавать, изменять, удалять и просматривать разделы. Поддерживаются следующие стандарты разметки дисков:
  - Master Boot Record (MBR) – она используется компьютерами на базе BIOS. Можно создавать основные, расширенные и логические разделы.
  - GUID Partition Table (GPT) – она использует Globally Unique Identifier (GUID) и предоставляет уникальные GUID дисков и разделов.

Опции потребления хранилища:

- Управление блочным хранилищем – данные хранятся в форме блоков, где каждый блок имеет уникальный идентификатор.
- Файловое хранилище – данные хранятся на файловом уровне на локальной системе. К этим данным можно получить локальный доступ при помощи файловых систем XFS (по умолчанию) или ext4 и по сети при помощи NFS и SMB.

Создание и управление логическими томами:

- Logical Volume Manager (LVM) – он создает логические устройства из физических устройств. Логические тома (LV) – это комбинация из физических томов (PV), объединенных в группы томов (VG). Настройка LVM состоит из следующих шагов:
  - Создания физических томов (PV) из жестких дисков.
  - Создания групп томов (VG) из физических томов (PV).
  - Создания логических томов (LV) из групп томов и назначения точек монтирования логическим томам (LV).

Локальные файловые системы:

- XFS.

- Ext2/3/4.

При помощи **parted** системный администратор может просмотреть таблицу разделов блочного устройства и подробную информацию об отдельных разделах.

Просмотр таблицы разделов при помощи parted.

Процедура:

1. Запустите интерактивную оболочку **parted**.

```
# parted block-device
```

   - Замените *block-device* на путь к устройству, которое хотите просмотреть, например, **/dev/sda**.
2. Выведите таблицу разделов.

```
(parted) print
```

Опционально, воспользуйтесь командой **select** для переключения к другому блочному устройству, которое нужно проверить следующим.

```
(parted) select block-device
```

Пример вывода команды **parted print**:

```
(parted) print
Модель: VMware, VMware Virtual S (scsi)
Диск /dev/sda: 21,5GB
Размер сектора (логич./физич.): 512B/512B
Таблица разделов: msdos
Флаги диска:

Номер  Начало  Конец   Размер  Тип      Файловая система  Флаги
 1     1049kB  1075MB  1074MB  primary  ext4              загрузочный
 2     1075MB  21,5GB  20,4GB  primary                    lvm
```

Вывод состоит из следующих полей:

Модель: VMware, VMware Virtual S (scsi)

- Тип диска, производитель, номер модели и интерфейс.

Диск /dev/sda: 21,5GB

- Путь к файлу блочного устройства и емкость хранилища.

Таблица разделов: msdos

- Тип заголовка диска.

Номер

- Номер раздела. Например, раздел с минорным номером 1, соответствует устройству /dev/sda1.

Начало и **Конец**

- Расположение на устройстве, где начинается и заканчивается раздел.

Тип

- Тип. Корректные типы: **metadata**, **free**, **primary**, **extended** или **logical**.

Файловая система

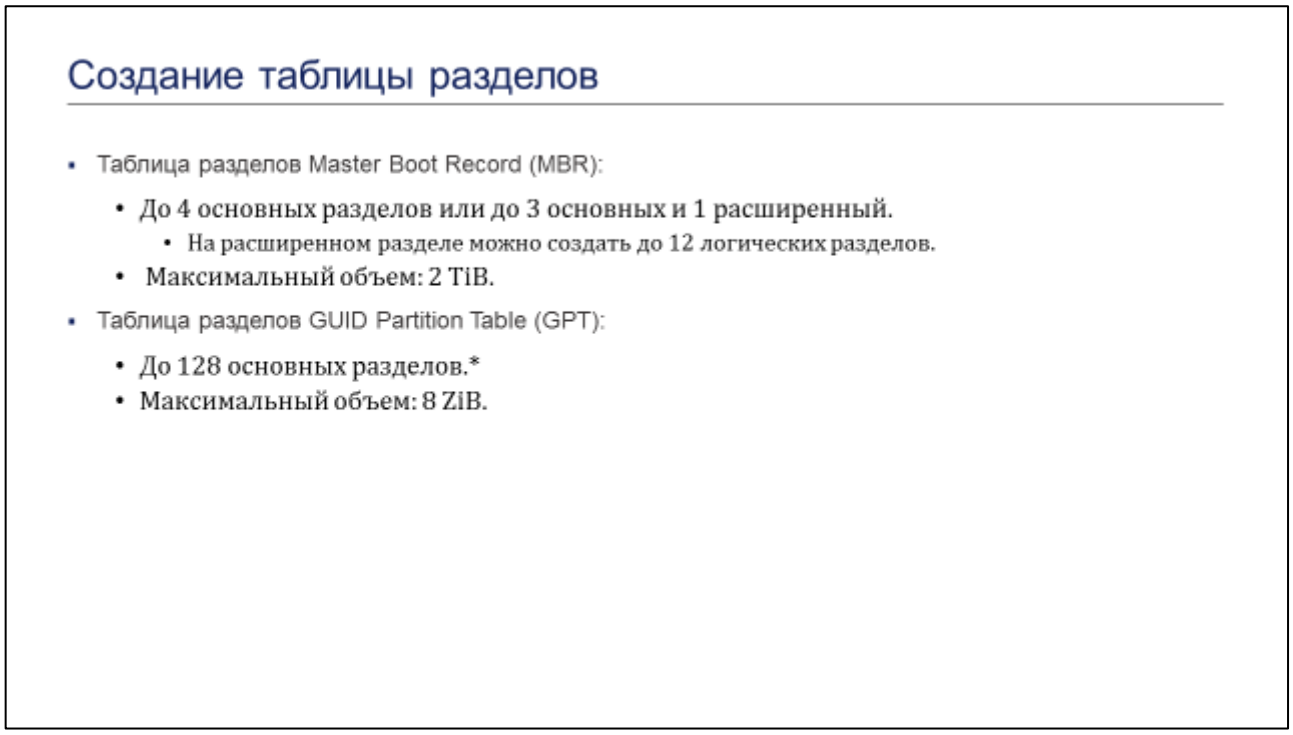
- Тип файловой системы. Если поле File system не показывает никакого значения, это обозначает, что файловая система не распознана. Утилита parted не может распознавать файловую систему на зашифрованных устройствах.

Флаги

- Список флагов, установленных для раздела. Доступные флаги: **boot**, **root**, **swap**, **hidden**, **raid**, **lvm** или **lba**.

Дополнительные ресурсы:

- Страница руководства **parted(8)**.



Системный администратор может форматировать блочное устройство при помощи различных типов таблиц разделов для обеспечения возможности использования разделов на устройстве.

ПРЕДУПРЕЖДЕНИЕ

Форматирование блочного устройства с таблицей разделов приводит к удалению всех данных, сохраненных на устройстве.

Рекомендации перед изменением таблицы разделов на диске.

Максимальное число разделов.

Число разделов на устройстве ограничено типом таблицы разделов.

- На устройствах с таблицей разделов Master Boot Record (MBR):
  - До 4 основных разделов (Primary Partition).
  - До трех основных разделов и один расширенный раздел (Extended Partition) с множеством логических разделов на нем.
- На устройства с таблицей разделов GUID Partition Table (GPT), максимальное число разделов – это 128. Несмотря на то, что спецификация GPT позволяет создавать больше разделов, за счет увеличения области, зарезервированной под таблицу разделов, общая практика, используемая утилитой **`parted`**, ограничивает возможности 128-ю разделами.

Максимальный размер раздела.

Размер раздела на устройстве ограничен типом таблицы разделов.

- На устройствах с таблицей разделов Master Boot Record (MBR) максимальный размер составляет **2 TiB**.
- На устройстве с таблицей разделов GUID Partition Table (GPT), максимальный размер составляет **8 ZiB**.

Выделение размера.

Утилита parted позволяет указывать размер раздела при помощи различных суффиксов:

1. **MiB**, **GiB** или **TiB** – размер вычисляется степенью числа 2. Точкой окончания является выделение до указанного размера минус 1 сектор.
2. **MB**, **GB** или **TB** – размер вычисляется степенью числа 10.

Точки начала и окончания вычисляются половиной указанной единицы, например, ±500KB при использовании суффикса **MB**.

Сравнение типов таблиц разделов.

Сравнение свойств различных типов таблиц разделов, которые можно создавать на блочном устройстве.

Таблица разделов	Максимальное число разделов	Максимальный размер раздела
Master Boot Record (MBR)	4 основных или 3 основных и 12 логических на расширенном	2TiB
GUID Partition Table (GPT)	128	8 ZiB

Создание таблицы разделов на диске при помощи `parted`.

Процедура:

1. Запустите интерактивную оболочку **`parted`**.

```
# parted block-device
```

2. Просмотрите информацию о разделе, чтобы выяснить имеются ли на устройстве разделы.

```
(parted) print
```

Если на устройстве уже имеются разделы, возможно, их потребуется удалить.

3. Создайте новую таблицу разделов.

```
(parted) mklabel table-type
```

Замените *`table-type`* на необходимый тип таблицы разделов:

   - **`msdos`** для MBR.
   - **`gpt`** для GPT.

Пример создания таблицы разделов GPT:

```
(parted) mklabel gpt.
```

Изменения вступят в силу сразу после ввода команды.

4. Убедитесь, что таблица разделов существует:

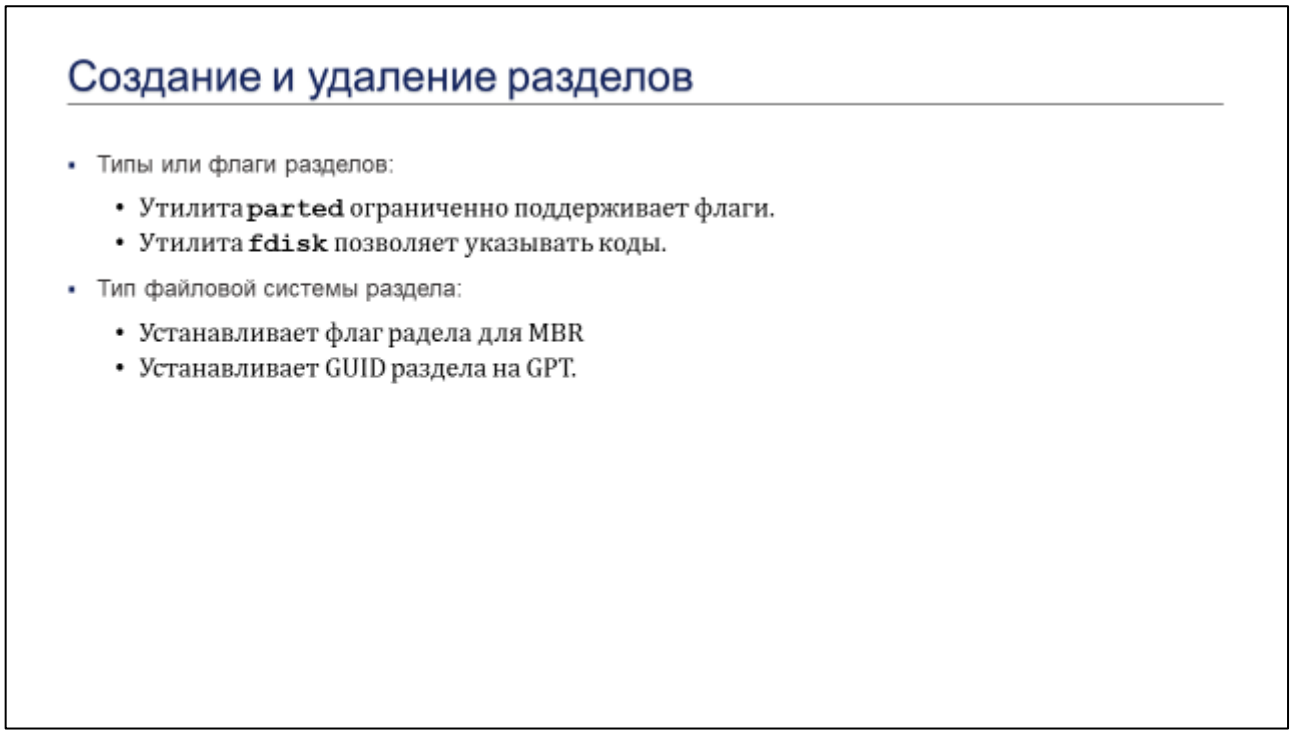
```
(parted) print
```

5. Выйдите из оболочки **parted**.

```
(parted) quit
```

Дополнительные ресурсы:

- Страница руководства **parted(8)**.



При помощи утилиты **parted** системный администратор может создавать новые разделы на диске.

Типы разделов.

Тип раздела или флаг, используется запущенной системой очень редко. Однако тип раздела важен для генераторов на лету, таких как **system-gpt-auto-generator**, которые используют тип раздела, например, для автоматического определения и монтирования устройств.

- Утилита **parted** предоставляет некоторое управление типами разделов за счет привязки типа разделов к флагу. Утилита **parted** может обрабатывать только определенные типы разделов, например, LVM, swap или RAID.
- Утилита **fdisk** поддерживает полный диапазон типов разделов за счет указания шестнадцатеричных кодов.

Тип файловой системы раздела.

Утилита **parted** опционально принимает аргумент типа файловой системы при создании раздела. Значение используется для:

- Установки флага раздела на MBR.
- Установки типа UUID раздела на GPT. Например, разные типы файловых систем, таких как **swap**, **fat** или **hfs** устанавливают различные GUID. Значение по умолчанию – это Linux Data GUID.

Аргумент не изменяет файловую систему на разделе в любом случае. Разница заключается только в поддерживаемых флагах или GUID.

Поддерживаются следующие типы файловых систем:

- **xfs**.
- **ext2**.
- **ext3**.
- **ext4**.
- **fat16**.
- **fat32**.
- **hfs**.
- **hfs+**.
- **linux-swap**.
- **ntfs**.
- **reiserfs**.

Создание раздела при помощи parted.

Предварительные требования:

- Наличие таблицы разделов на диске.
- Если создаваемый раздел будет размером больше 2 TiB, диск должен быть отформатирован при помощи GUID Partition Table (GPT).

Процедура:

1. Запустите интерактивную оболочку parted.

```
# parted block-device
```

2. Просмотрите текущую таблицу разделов для определения свободного пространства.

```
(parted) print
```

3. Если не хватает свободного места, потребуется удалить не нужные разделы.
4. Из таблицы разделов потребуется следующая информация:
   - Точка начала и окончания нового раздела.
   - На диске MBR должен быть тип раздела.
5. Создайте новый раздел.

```
(parted) mkpart part-type name fs-type start end
```

   - Замените *part-type* необходимым типом раздела: primary, logical или extended. Это применимо только для таблицы разделов MBR.
   - Замените *name* на подходящее имя раздела. Это необходимо для таблицы разделов GPT.
   - Замените *fs-type* одним из значений *xfs*, *ext2*, *ext3*, *ext4*, *fat16*, *fat32*, *hfs*, *hfs+*, *linux-swap*, *ntfs* или *reiserfs*.
   - Замените *start* и *end* размером, который определяет начальную и конечную точки раздела, при этом можно использовать суффиксы при указании раздела (**512MiB**, **20GiB**, **1TiB**). По умолчанию размер указывается в мегабайтах.

Пример создания основного раздела с **1024 MiB** до **2048 MiB** на таблице разделов MBR:

```
(parted) mkpart primary 1024MiB 2048MiB
```

Изменения вступят в силу сразу после ввода команды.

6. Просмотрите таблицу разделов, чтобы убедиться, что раздел успешно создан с корректным типом раздела, типом файловой системы и размером:

```
(parted) print
```

7. Выйдите из оболочки **parted**.

```
(parted) quit
```

8. Воспользуйтесь командой **udevadm** чтобы дождаться, пока система зарегистрирует новый узел устройства.

```
# udevadm settle
```

9. Убедитесь, что ядро распознало новый раздел.

```
# cat /proc/partitions
```

Дополнительные ресурсы:

- Страница руководства **parted(8)**.

Установка типа раздела при помощи утилиты fdisk.

Системный администратор может установить тип раздела или флаг при помощи утилиты **fdisk**.

Предварительные требования:

- Раздел на диске.

Процедура:

1. Запустите интерактивную оболочку **fdisk**.

```
# fdisk block-device
```

2. Просмотрите текущую таблицу разделов, чтобы определить минорный номер раздела.

```
Command (m for help): print
```

Текущий тип раздела можно посмотреть в колонке «**Type**», а идентификатор – в колонке «**Id**».

3. Введите команду **type** и выберите номер раздела.

```
Command (m for help): type
Partition number (1,2,3 default 3): 2
```

4. Опционально, выведите список доступных шестнадцатеричных кодов.

```
Hex code (type L to list all codes): L
```

5. Установите тип раздела.

```
Hex code (type L to list all codes): 8e
```

6. Запишите изменения на диск и выйдите из оболочки **fdisk**.

```
Command (m for help): write
The partition table has been altered.
Syncing disks.
```

7. Убедитесь, что изменения вступили в силу.

```
# fdisk --list block-device
```

Удаление раздела.

Системный администратор может удалять разделы диска, которые более не нужны для освобождения дискового пространства.

ПРЕДУПРЕЖДЕНИЕ

Удаление раздела приводит к удалению всех данных, сохраненных на разделе.

Процедура:

1. Запустите интерактивную оболочку **parted**.

```
# parted block-device
```

2. Просмотрите текущую таблицу разделов для определения минорного номера раздела для удаления.

```
(parted) print
```

3. Удалите раздел.

```
(parted) rm minor-number
```

Замените *minor-number* минорным номером раздела для удаления, например – **3**.

Изменения вступят в силу сразу после ввода команды.

4. Убедитесь, что раздел успешно удален из таблицы разделов.

```
(parted) print
```

6. Выйдите из оболочки **parted**.

```
(parted) quit
```

7. Убедитесь, что ядро знает об изменения таблицы разделов диска.

```
# cat /proc/partitions
```

8. Удалите раздел из файла **/etc/fstab**, если раздел был настроен на автоматическое монтирование.
9. Восстановите юниты монтирования (Mount Units), чтобы система зарегистрировала новую конфигурацию системы:

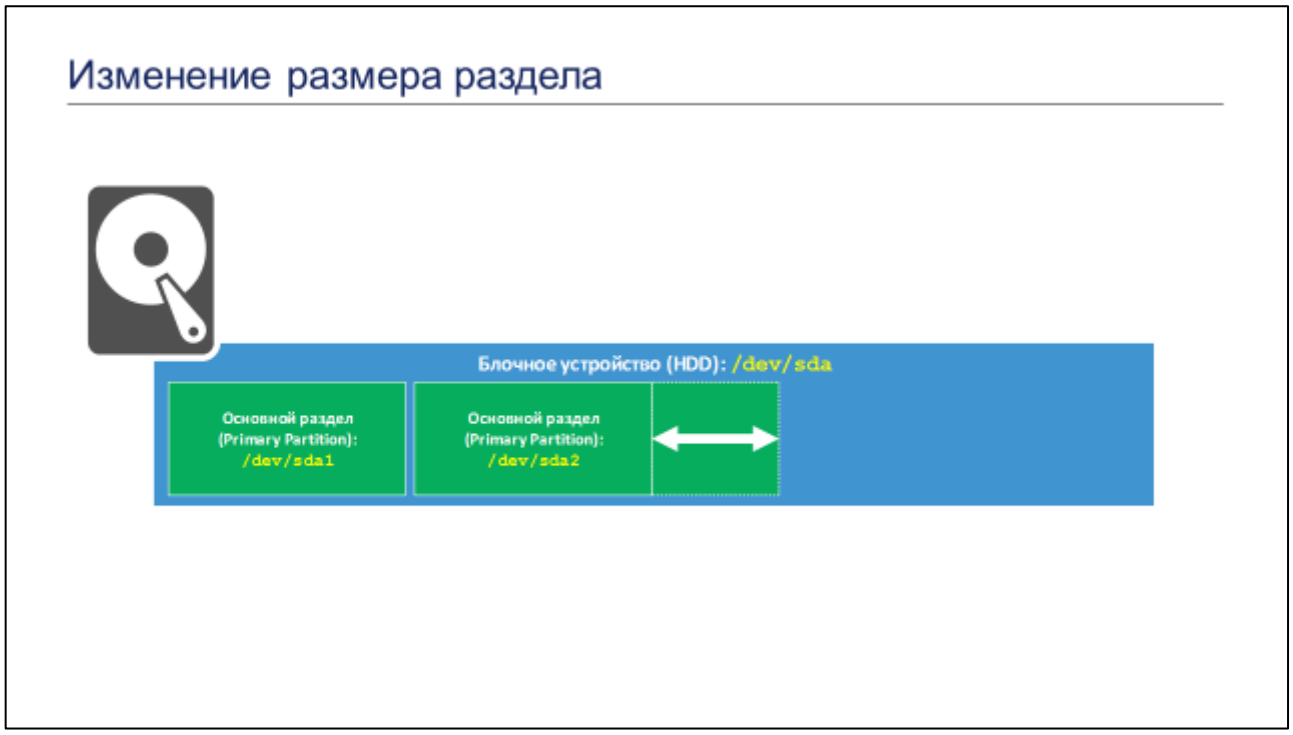
```
# systemctl daemon-reload
```

10. Если был удален раздел подкачки или удалены части LVM, удалите все ссылки на раздел из командной строки ядра в файле **/etc/default/grub** и восстановите конфигурацию GRUB.
11. Для регистрации изменений в начальной загрузке системы, пересоздайте файловую систему **initramfs**:

```
# dracut --force --verbose
```

Дополнительные ресурсы:

- Страница руководства **parted(8)**.



Системный администратор может увеличивать за счет неиспользованного пространства диска или уменьшать раздел при помощи parted.

Изменение размера раздела при помощи `parted`.

Предварительные требования:

- Перед сжатием раздела, необходимо выполнить резервное копирование всех данных на разделе.

Процедура:

1. Если необходимо сжать раздел, необходимо предварительно выполнить сжатие файловой системы на нем. Обратите внимание, что файловая система XFS не поддерживает сжатие.
3. Запустите интерактивную оболочку **`parted`**.

```
# parted block-device
```

4. Просмотрите текущую таблицу разделов.

```
(parted) print
```

Из таблицы разделов определите:

- Минорный номер раздела.
- Расположение существующего раздела и его новую точку окончания после изменения раздела.

5. Измените размер раздела.

```
(parted) resizepart minor-number new-end
```

- Замените *minor-number* на минорный номер раздела, который необходимо изменить, например, 3.
- Замените *new-end* размером, который определяет новую точку окончания для измененного раздела (отсчет берется от начала диска). При указании размер можно использовать суффиксы, например, **512MiB**, **20GiB** или **1.5TiB**. По умолчанию размер указывается в мегабайтах.

Пример увеличения раздела, расположенного в начале диска до **2GiB**:

```
(parted) resizepart 1 2GiB
```

Изменения вступают в силу сразу после ввода команды.

6. Просмотрите таблицу разделов, чтобы убедиться, что размер раздела успешно изменен.

```
(parted) print
```

7. Выйдите из оболочки parted.

```
(parted) quit
```

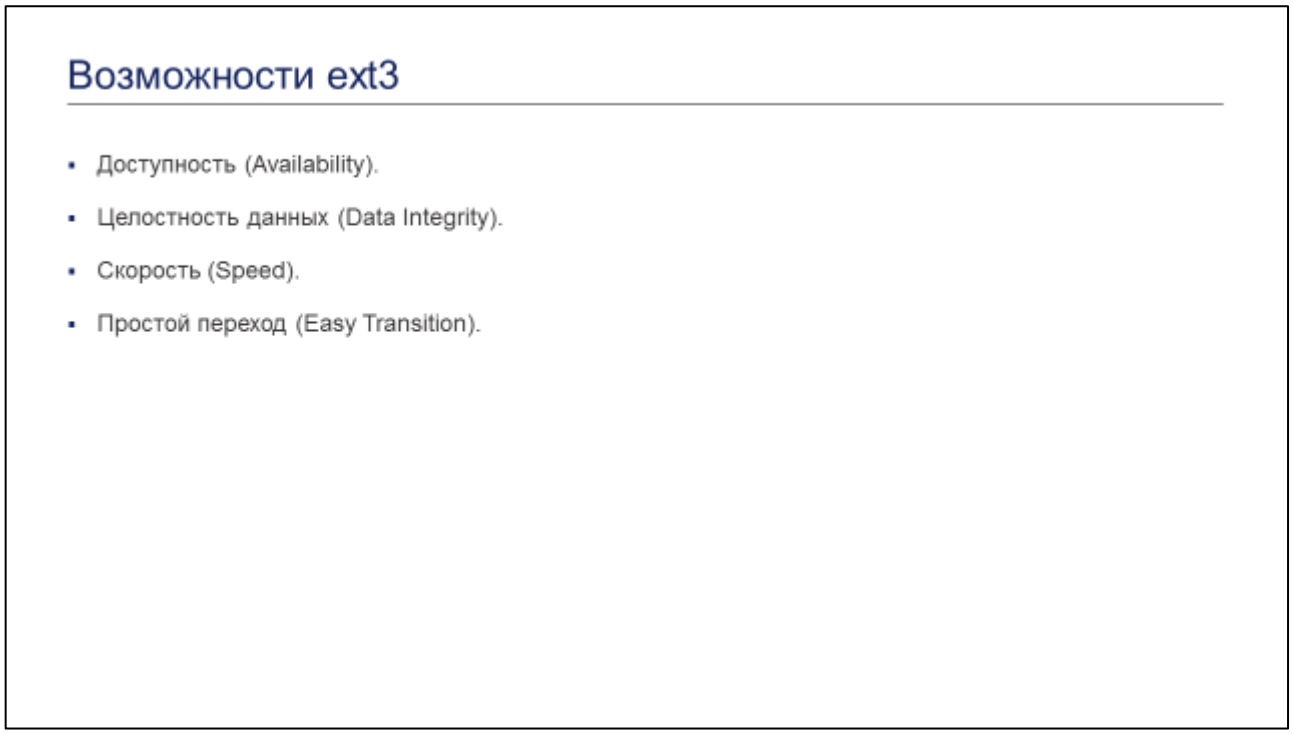
8. Убедитесь, что ядро распознало новый раздел.

```
# cat /proc/partitions
```

9. После изменения размера раздела, увеличьте файловую систему на нем.

Дополнительные ресурсы:

- Страница руководства **parted(8)**.



Системный администратор может создавать, монтировать, изменять размер и восстанавливать файловую систему ext3. Файловая система ext3 в основе своей – расширенная версия файловой системы ext2.

Возможности файловой системы ext3:

- **Доступность (Availability)**: после непредвиденного отключения электроэнергии или аварии системы, проверка файловой системы не требуется, так как файловая система поддерживает журналирование. Размер журнала по умолчанию требует нескольких секунд на восстановление в зависимости от скорости оборудования.

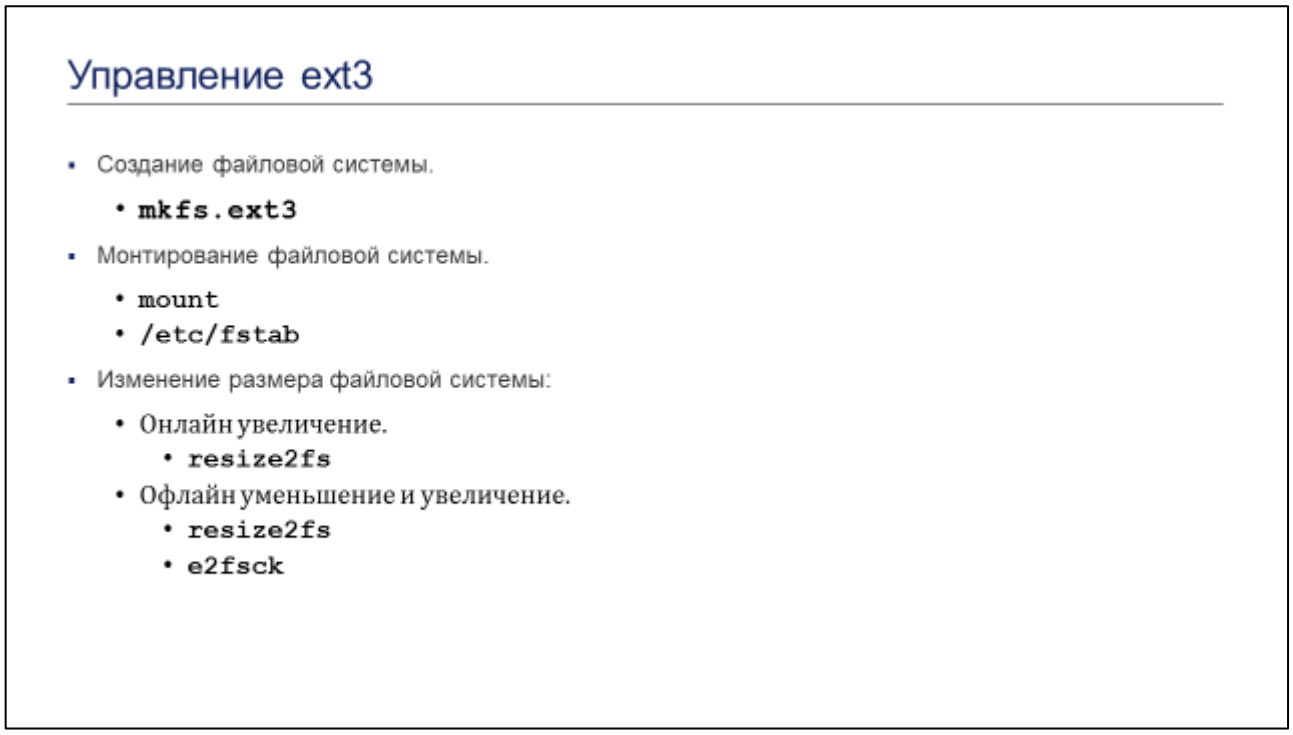
ПРИМЕЧАНИЕ

Единственный поддерживаемый режим журналирования на файловой системе ext3 – **`data=ordered (default)`**.

- **Целостность данных (Data Integrity)**: Файловая система предотвращает потерю данных в случае непредвиденного отключения электроэнергии или аварии системы.
- **Скорость (Speed)**: Не смотря на запись некоторых данных более одного раза, в большинстве случаев ext3 обеспечивает большую пропускную способность, чем ext2, за счет оптимизации движения головки накопителя при помощи журнала.
- **Простой переход (Easy Transition)**: Возможность простой миграции с файловой системы ext2 на файловую систему ext3 и получение преимуществ от надежной журналируемой файловой системы без форматирования.

Дополнительные ресурсы:

- Страница руководства **`ext3`**.



Системный администратор может создать файловую систему ext3 на блочном устройстве при помощи команды **mkfs.ext3**.

Предварительные требования:

- Раздел на диске, логический том или программный RAID-массив.

Процедура:

1. Для создания файловой системы ext3 на устройстве с обыкновенными разделами, томе LVM/MD или подобных устройствах, воспользуйтесь следующей командой:

```
# mkfs.ext3 /dev/block_device
```

Замените **/dev/***block_device* на путь к блочному устройству.

Например, **/dev/sdb1**, **/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a** или **/dev/my-volgroup/my-lv**. Как правило, опции по умолчанию оптимальны для большинства сценариев.

ПРИМЕЧАНИЕ

Для указания UUID при создании файловой системы необходимо использовать опцию **-U**:

```
mkfs.ext -U UUID /dev/block_device
```

Замените *UUID* на необходимый набор UUID, например, **7cd65de3-e0be-41d9-b66d-96d749c02da7**.

ПРИМЕЧАНИЕ

Для указания заголовка (Label) при создании файловой системы необходимо использовать опцию **-L**:

```
mkfs.ext -L label-name /dev/block_device
```

2. Для просмотра созданной файловой системы:

```
# blkid
```

Дополнительные ресурсы:

- Страница руководства **ext3**.
- Страница руководства **mkfs.ext3**.

Монтирование файловой системы ext3.

Системный администратор может монтировать файловую систему ext3 при помощи утилиты **mount**.

Предварительные требования:

- Файловая система ext3.

Процедура:

1. Создайте точку монтирования для файловой системы:

```
# mkdir /mount/point
```

Замените */mount/point* на имя каталога, к которому необходимо примонтировать раздел.

2. Смонтируйте файловую систему ext3 (файловую систему можно монтировать без указания каких-либо дополнительных опций):

```
# mount /dev/block_device /mount/point
```

3. Для просмотра смонтированной файловой системы:

```
# df -h
```

Дополнительные ресурсы:

- Страница руководства **ext3**.
- Страница руководства **mount**.
- Страница руководства **fstab**.

Изменение размера файловой системы ext3.

Системный администратор может изменять размер файловой системы ext3 при помощи утилиты **resize2fs**. Утилита **resize2fs** читает размер в единицах размера блока файловой системы, если не указан суффикс, определяющий единицы:

- **s** (sectors) – секторы (512 байт).
- **K** (kilobytes) – килобайты (1,024 байта).
- **M** (megabytes) – мегабайты (1,048,576 байт).
- **G** (gigabytes) – гигабайты (1,073,741,824 байта).
- **T** (terabytes) – терабайты (1,099,511,627,776 байт).

Предварительные требования:

- Файловая система ext3.
- Блочное устройство подходящего размера, для хранения файловой системы после изменения размера.

Процедура:

1. Для изменения размера файловой системы ext3, воспользуйтесь следующими шагами:
   A. Для увеличения или уменьшения файловой системы в режиме офлайн:

```
# umount /dev/block_device
# e2fsck -f /dev/block_device
# resize2fs /dev/block_device size
```

   - Замените **/dev/***block_device* на путь к блочному устройству, например **/dev/sdb1**.
   - Замените *size* необходимым значением измененного размера при помощи суффиксов **s**, **K**, **M**, **G** и **T**.

   B. Файловая система ext3 может быть увеличена смонтированной, при помощи команды **resize2fs**:

```
# resize2fs /mount/device size
```

ПРИМЕЧАНИЕ

Параметра указывающий размер – опционален и может быть пропущен при увеличении. Утилита **resize2fs** автоматически увеличит файловую систему для заполнения всего доступного объема блочного устройства.

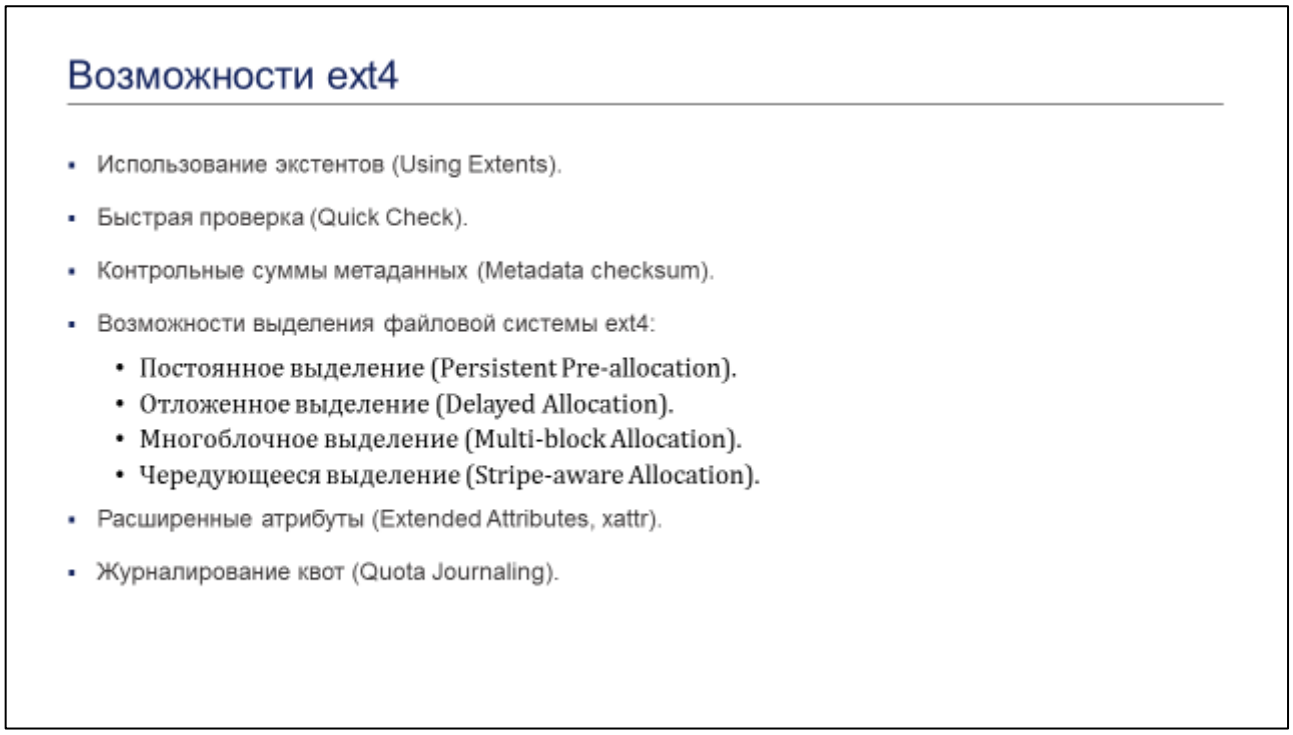
2. Для просмотра измененной файловой системы:

```
# df -h
```

Дополнительные ресурсы:

- Страница руководства **resize2fw**.

- Страница руководства **`e2fsck`**.
- Страница руководства **`ext3`**.



Системный администратор может создавать, монтировать, изменять размер, выполнять резервное копирование и восстанавливать файловую систему ext4. Файловая система ext4 в основе своей – это масштабируемое расширение файловой системы ext3. В CentOS 8 (RHEL 8) файловая система ext4 максимально поддерживает размер в 50 TiB с максимальным размером отдельного файла – 16 TiB.

Возможности файловой системы ext4:

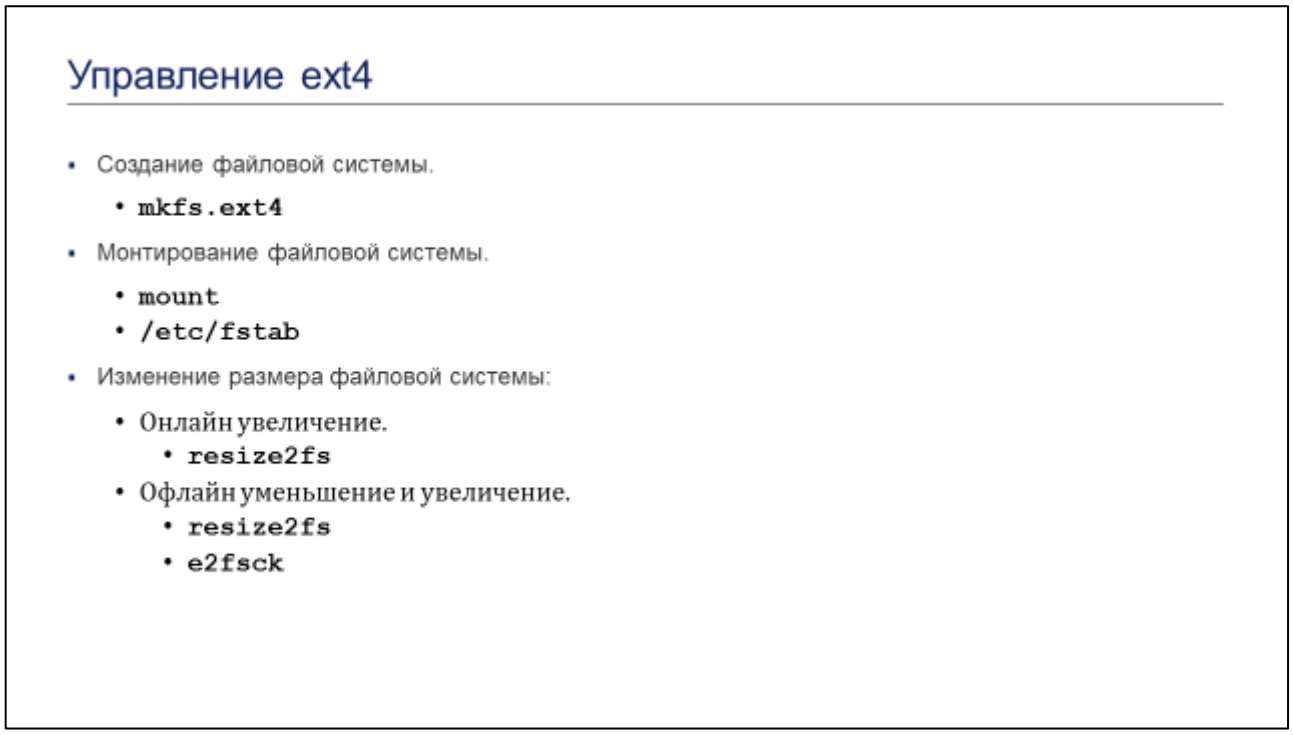
- **Использование экстентов (Using Extents)**: Файловая система ext4 использует экстенты, которые улучшают производительность при использовании больших файлов и сокращает накладные расходы метаданных для больших файлов.
- **Быстрая проверка (Quick Check)**: Ext4 маркирует неразмеченные группы блоков и сектора таблицы инодов (Inode), что позволяет пропускать группы блоков и секторы таблицы в процессе проверки файловой системы. Это приводит к быстрой проверке файловой системы и преимуществам по мере роста файловой системы.
- **Контрольные суммы метаданных (Metadata checksum)**: По умолчанию данная возможность включена в CentOS 8 (RHEL 8).
- Возможности выделения файловой системы ext4:
    - Постоянное выделение (Persistent pre-allocation).
    - Отложенное выделение (Delayed allocation).
    - Многоблочное выделение (Multi-block allocation).
    - Чередующееся выделение (Stripe-aware allocation).
- **Расширенные атрибуты (Extended attributes, `xattr`)**: Это позволяет системе связывать несколько дополнительных пар имен и значений с файлами.
- **Журналирование квот (Quota journaling)**: Это позволяет избегать необходимости длительных проверок целостности квот после аварий.

ПРИМЕЧАНИЕ

Единственный поддерживаемый режим журналирования на файловой системе ext3 – **`data=ordered (default)`**.

Дополнительные ресурсы:

- Страница руководства **`ext4`**.



Системный администратор может создать файловую систему ext4 на блочном устройстве при помощи команды **mkfs.ext4**.

Предварительные требования:

- Раздел на диске, логический том или программный RAID-массив.

Процедура:

1. Для создания файловой системы ext4:
- На устройстве с обыкновенными разделами, томе LVM/MD или подобных устройствах:

```
# mkfs.ext4 /dev/block_device
```

Замените **/dev/***block_device* на путь к блочному устройству.

Например, **/dev/sdb1**, **/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a** или **/dev/my-volgroup/my-lv**. Как правило, опции по умолчанию оптимальны для большинства сценариев.

- Для чередующихся блочных устройств (например, массивов RAID5), чередующаяся геометрия может быть указана во время создания файловой системы. Использование корректной геометрии увеличивает производительность файловой системы ext4. Например, для создания файловой системы с шагом 64k (16 раз по 4096) на файловой системе с блоком в 4k, воспользуйтесь следующей командой:

```
# mkfs.ext4 -E stride=16,stripe-width=64 /dev/block_device
```

В данном примере:

- **stride=***value*: Указывает размер блока (Chunk Size) RAID-массива.
- **stripe-width=***value*: Указывает число дисков с данными в RAID-массиве умноженное на размер блока RAUD-массива.

ПРИМЕЧАНИЕ

Для указания UUID при создании файловой системы необходимо использовать опцию **-U**:

```
mkfs.ext -U UUID /dev/block_device
```

Замените *UUID* на необходимый набор UUID, например, **7cd65de3-e0be-41d9-b66d-96d749c02da7**.

ПРИМЕЧАНИЕ

Для указания заголовка (Label) при создании файловой системы необходимо использовать опцию **-L**:

```
mkfs.ext -L label-name /dev/block_device
```

2. Для просмотра созданной файловой системы:

```
# blkid
```

Дополнительные ресурсы:

- Страница руководства **ext4**.
- Страница руководства **mkfs.ext4**.

Монтирование файловой системы ext4.

Системный администратор может монтировать файловую систему ext4 при помощи утилиты mount.

Предварительные требования:

- Файловая система **ext4**.

Процедура:

1. Создайте точку монтирования для файловой системы:

```
# mkdir /mount/point
```

Замените */mount/point* на имя каталога, к которому необходимо примонтировать раздел.

2. Смонтируйте файловую систему ext4 (файловую систему можно монтировать без указания каких-либо дополнительных опций):

```
# mount /dev/block_device /mount/point
```

3. Для просмотра смонтированной файловой системы:

```
# df -h
```

Дополнительные ресурсы:

- Страница руководства **ext4**.
- Страница руководства **mount**.
- Страница руководства **fstab**.

Изменение размера файловой системы ext4.

Системный администратор может изменять размер файловой системы ext4 при помощи утилиты **resize2fs**. Утилита **resize2fs** читает размер в единицах размера блока файловой системы, если не указан суффикс, определяющий единицы:

- **s** (sectors) – секторы (512 байт).
- **K** (kilobytes) – килобайты (1,024 байта).
- **M** (megabytes) – мегабайты (1,048,576 байт).
- **G** (gigabytes) – гигабайты (1,073,741,824 байта).
- **T** (terabytes) – терабайты (1,099,511,627,776 байт).

Предварительные требования:

- Файловая система ext4.
- Блочное устройство подходящего размера, для хранения файловой системы после изменения размера.

Процедура:

1. Для изменения размера файловой системы ext4, воспользуйтесь следующими шагами:
   a. Для увеличения или уменьшения файловой системы в режиме офлайн:

```
# umount /dev/block_device
# e2fsck -f /dev/block_device
# resize2fs /dev/block_device size
```

   - Замените */dev/block_device* на путь к блочному устройству, например, **/dev/sdb1**.
   - Замените *size* необходимым значением измененного размера при помощи суффиксов **s**, **K**, **M**, **G** и **T**.

   b. Файловая система ext4 может быть увеличена смонтированной, при помощи команды **resize2fs**:

```
# resize2fs /mount/point size
```

ПРИМЕЧАНИЕ

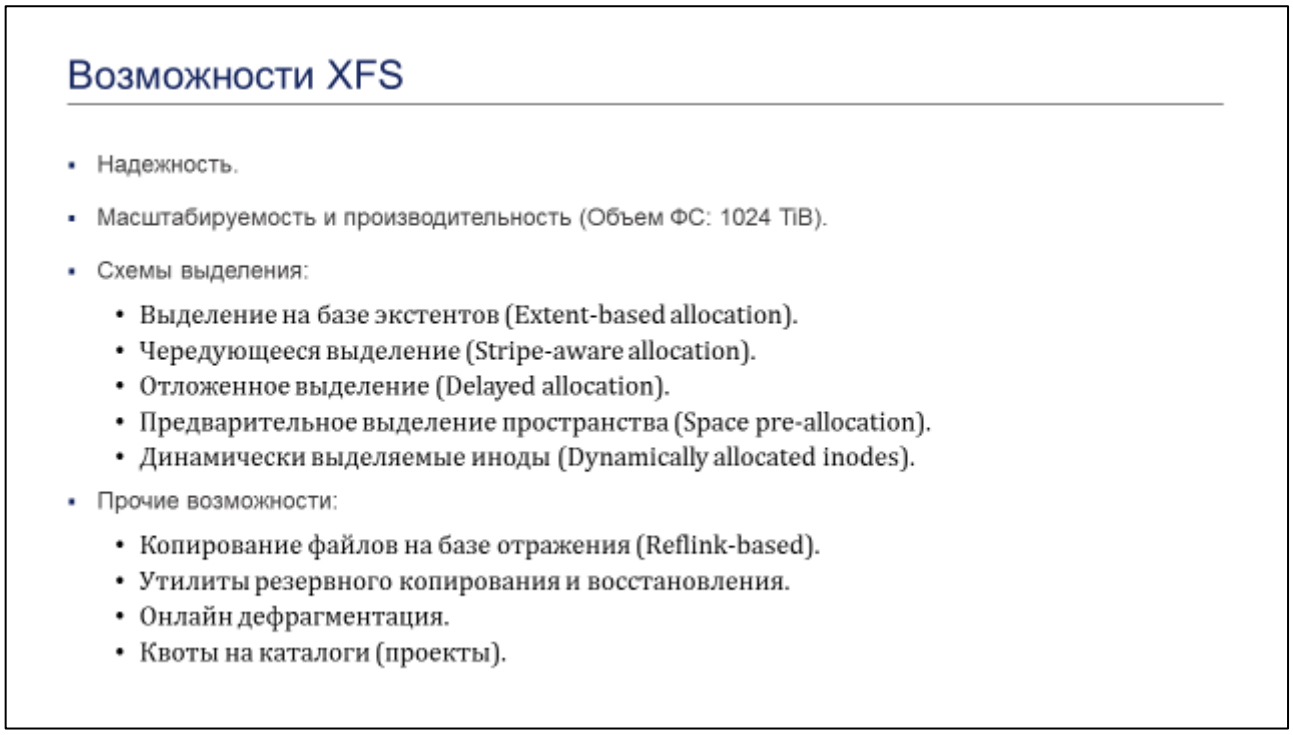
Параметра указывающий размер – опционален и может быть пропущен при увеличении. Утилита **resize2fs** автоматически увеличит файловую систему для заполнения всего доступного объема блочного устройства.

2. Для просмотра измененной файловой системы:

```
# df -h
```

Дополнительные ресурсы:

- Страница руководства **resize2fw**.
- Страница руководства **e2fsck**.
- Страница руководства **ext4**.



XFS – это высоко масштабируемая, высоко производительная, надежная и зрелая 64-ех битная журналируемая файловая система, которая поддерживает очень большие файлы и файловые системы на одном хосту. Файловая система XFS – файловая система по умолчанию в CentOS 7 (RHEL 7) и CentOS 8 (RHEL 8). Изначально XFS была разработана в начале 1990-ых годов компанией SGI. XFS имеет длительную историю применения на экстремально больших серверах и массивах хранилища.

Возможности XFS:

- Надежность:
  - Журналирование метаданных, которое обеспечивает целостность файловой системы после аварий системы, за счет сохранения записей операций файловой системы, которые могут быть воспросизведены, когда система будет повторно запущена, а файловая система вновь смонтирована.
  - Интенсивная проверка целостности метаданных во время работы.
  - Масштабируемые и быстрые утилиты восстановления.
  - Журналирование квот. Это позволяет избегать необходимости длительных проверок целостности квот после аварий.
- Масштабируемость и производительность:
  - Поддерживаемый объем файловой системы равен 1024 TiB.
  - Возможность поддерживать большое число конкурентных операций.
  - Индексирование B-дерева (B-Tree) для масштабирования управления свободным пространством.
  - Интеллектуальные алгоритмы предварительного чтения метаданных.
  - Оптимизации для рабочих нагрузок поточного видео.
- Схемы выделения:
  - Выделение на базе экстентов (Extent-based allocation).
  - Чередующееся выделение (Stripe-aware allocation).
  - Отложенное выделение (Delayed allocation).
  - Предварительное выделение пространства (Space pre-allocation).

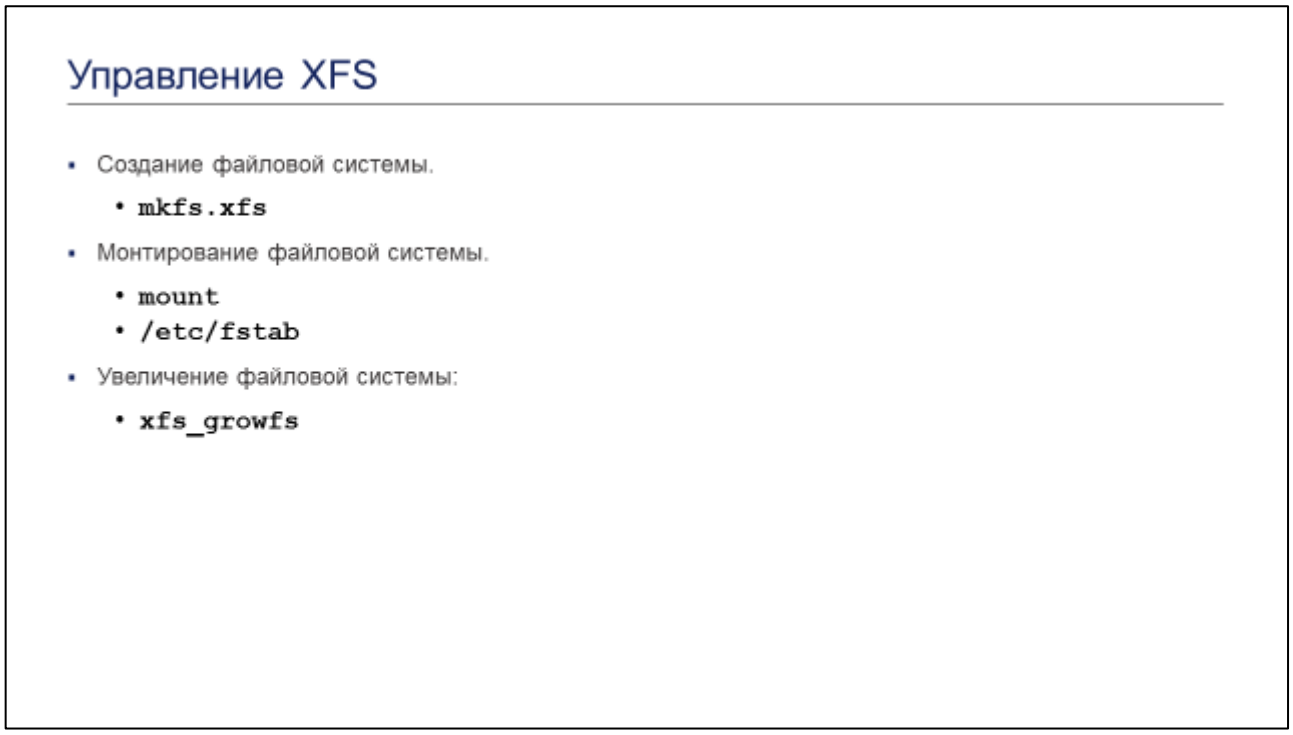
    - Динамически выделяемые иноды (Dynamically allocated inodes).
- Прочие возможности:
    - Копирование файлов на базе отражения (Reflink-based) [Новое в CentOS 8 (RHEL 8)].
    - Тесно интегрированные утилиты резервного копирования и восстановления.
    - Онлайн дефрагментация.
    - Увеличение файловых систем онлайн.
    - Обширные возможности диагностики.
    - Расширенные атрибуты (xattr). Это позволяет системе связывать несколько дополнительных пар имя/значение с файлом.
    - Квоты на каталоги (проекты). Это обеспечивает возможность применения ограничения квотами на дерево каталогов.
    - Штампы времени в долях секунды (Subsecond).

Характеристики производительности.

Файловая система XFS имеет высокую производительность на больших системах с корпоративными рабочими нагрузками. Большие системы, как правило обладают большим числом процессоров, HBA и соединений с дисковыми массивами.

Файловая система XFS также хорошо работает на системах меньшего размера, которые имеют многопоточные рабочие нагрузки с параллельным вводом/выводом.

Файловая система XFS обладает значительно более низкой производительностью для однопоточных рабочих нагрузок с интенсивным использованием метаданных, например, рабочие нагрузки, которые создают или удаляют большое число маленьких файлов в один поток.



Системный администратор может создавать файловую систему на блочном устройстве, чтобы иметь возможность сохранять файлы и каталоги.

Процедура:

1. Для создания файловой системы:
    a. Если устройство – является обычным разделом, томом LVM или MD, диском или подобным устройством, воспользуйтесь командой:

```
# mkfs.xfs block-device
```

Замените *block-device* на путь к блочному устройству. Например, **/dev/sdb1**, **/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a** или **/dev/myvolgroup/my-lv**.

Как правило, опции по умолчанию оптимальны для большинства сценариев.

При использовании **mkfs.xfs** на блочном устройстве, содержащем существующую файловую систему, необходимо добавить опцию **-f** для перезаписи файловой системы.

Для создания файловой системы на аппаратном устройстве RAID, необходимо проверить, что система корректно определяет чередующуюся геометрию устройства:

- Если информация о геометрии корректно, дополнительные опции при создании не потребуются:

```
# mkfs.xfs block-device
```

- Если информация не корректно, можно указать чередующуюся геометрию вручную, при помощи опции **-d** и ее параметров **su** и **sw**. Параметр **su** – указывает размер блока (Chink Size) RAID-массива, а параметр **sw** указывает число дисков с данными в RAID-массиве.

Пример:

```
# mkfs.xfs -d su=64k,sw=4 /dev/sda3
```

2. Убедитесь, что система зарегистрировала новый узел устройства.

```
# udevadm settle
```

Дополнительные ресурсы:

- Страница руководства **mkfs.xfs(8)**.

Увеличение размера файловой системы XFS.

Системный администратор может увеличивать размер файловой системы XFS, для утилизации большего объема хранилища.

ПРИМЕЧАНИЕ

На данный момент невозможно уменьшать размер файловой системы XFS.

Предварительные требования:

- Фоновое блочное устройство, обладающее соответствующим размером для хранения измененного размера файловой системы.
- Смонтированная файловая система XFS.

Процедура:

Для увеличения размер смонтированной файловой системы XFS воспользуйтесь командой **xfs_growfs**.

```
# xfs_growfs file-system -D new-size
```

Замените *file-system* на точку монтирования файловой системы XFS.

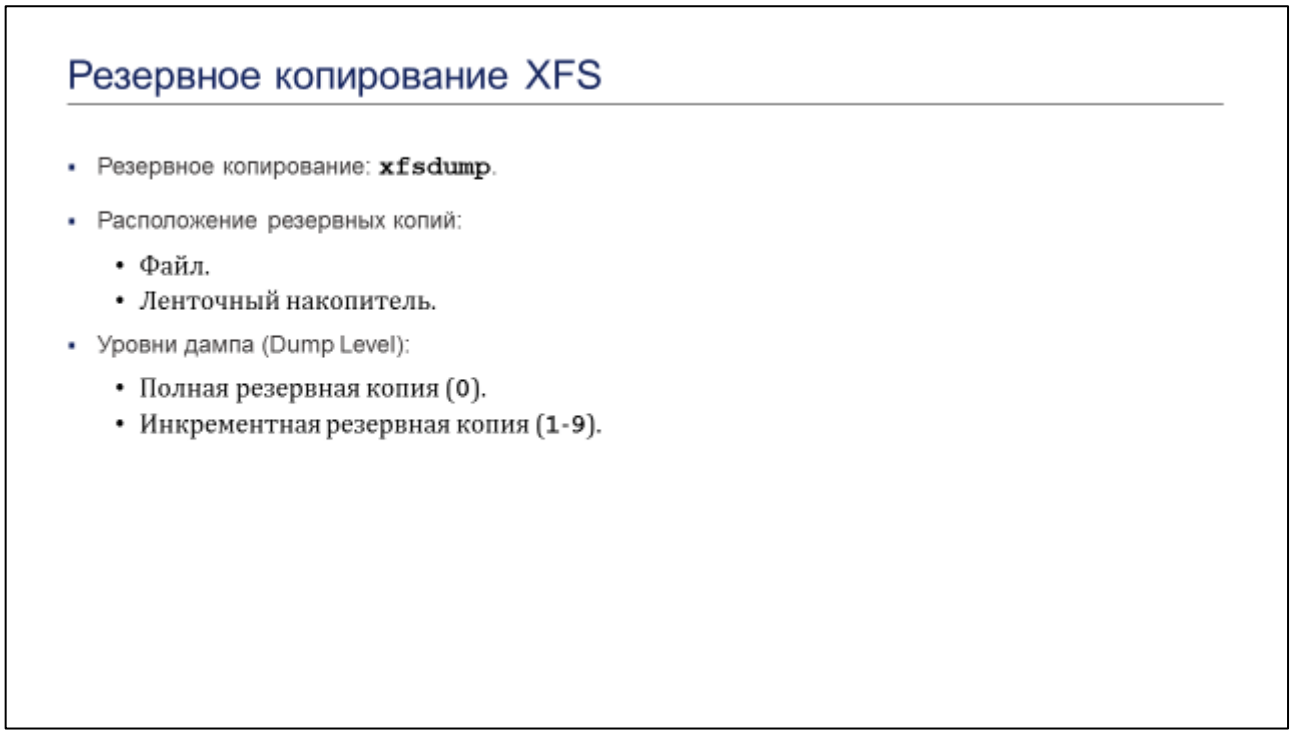
При использовании опции **-D**, Замените *new-size* на необходимый размер файловой системы, указанный в блоках файловой системы. Для определения размера блока в Kb на указанной файловой системе, воспользуйтесь утилитой **xfs_info**.

```
# xfs_info block-device
```

Без опции **-D**, **xfs_growfs** увеличивает файловую систему до размеров фонового устройства.

Дополнительные ресурсы:

- Страница руководства **xfs_growfs(8)**.



Системный администратор может использовать утилиту **xfsdump** для выполнения резервного копирования файловой системы в файл или на ленту. Это обеспечивает простой механизм резервного копирования.

При помощи утилиты **xfsdump** можно:

- В**ыполнять резервное копирование в обычный файл образов.** Только одна резервная копия может быть записана в обычный файл.
- **Выполнять резервное копирование на ленточный накопитель.** Утилита xfsdump также поддерживает возможность записи нескольких резервных копий на одну ленту. Резервная копия может разделяться на несколько лент.

Для резервного копирования нескольких файловых систем на одно ленточное устройство, достаточно просто записать резервную копию на ленту, которая уже содержит резервную копию XFS. Это добавит новую резервную копию к предыдущей. По умолчанию **xfsdump** никогда не переписывает существующие резервные копии.

- **Создавать инкрементные резервные копии.** Утилита **xfsdump** использует уровни дампа (Dump Levels) для определения базовой резервной копии к которой относятся остальные резервные копии.

Число от **0** до **9** относится к увеличивающемуся уровню дампа. Инкрементальная резервная копия содержит только те файлы, которые изменились с момента предыдущего дампа с меньшим уровнем:

  - Для создания полной резервной копии, необходимо сделать дамп с уровнем **0** на файловой системе.
  - Дамп первого уровня – это первая инкрементальная копия после полной резервной копии, следующая инкрементальная резервная копия будет иметь вровень **2**, это значит резервная копия будет содержать только те файлы, которые изменились с момента создания дампа с уровнем **1**. И так далее до максимального уровня **9**.

- Исключать файлы из резервной копии, при помощи размера, пути или флагов инода (Inode) для фильтрации их.

Дополнительные ресурсы:

- Страница руководства **xfsdump(8)**.

Резервное копирование файловой системы при помощи xfsdump.

Системный администратор может выполнять резервное копирование содержимого файловой системы в файл или на ленту.

Предварительные требования:

- Файловая система XFS для выполнения резервного копирования.
- Другая файловая система или ленточный накопитель для хранения резервной копии.

Процедура:

Для выполнения резервного копирования файловой системы XFS, воспользуйтесь следующей командой:

```
# xfsdump -l level [-L label] -f backup-destination path-to-xfs-filesystem
```

- Замените *level* на уровень дампа резервной копии. Используйте уровень 0 для создания полной резервной копии или от **1** до **9** для выполнения инкрементальных резервных копий.
- Замените *backup-destination* на путь, где необходимо поместить резервную копию. Точкой назначения может быть обычный файл, ленточный накопитель или удаленный ленточный накопитель. Например, **/backup-files/Data.xfsdump** для файла или **/dev/st0** для ленточного накопителя.
- Замените *path-to-xfs-filesystem* наточку монтирования файловой системы XFS, резервное копирование которой необходимо выполнить. Например, **/mnt/data/** (Файловая система должна быть смонтирована).

При резервном копировании нескольких файловых систем и сохранении их на одном ленточном устройстве, необходимо добавить заголовок сессии (Session Label) к каждой резервной копии, при помощи опции **-L** *label*, таким образом, будет проще идентифицировать их при восстановлении. Замените *label* на имя для резервной копии, например, **backup_data**.

Пример резервного копирования нескольких файловых систем XFS:

1. Резервное копирование содержимого файловых систем XFS, смонтированного в каталоги /boot и /data и сохранения их в файлы в каталоге /backup-files/:

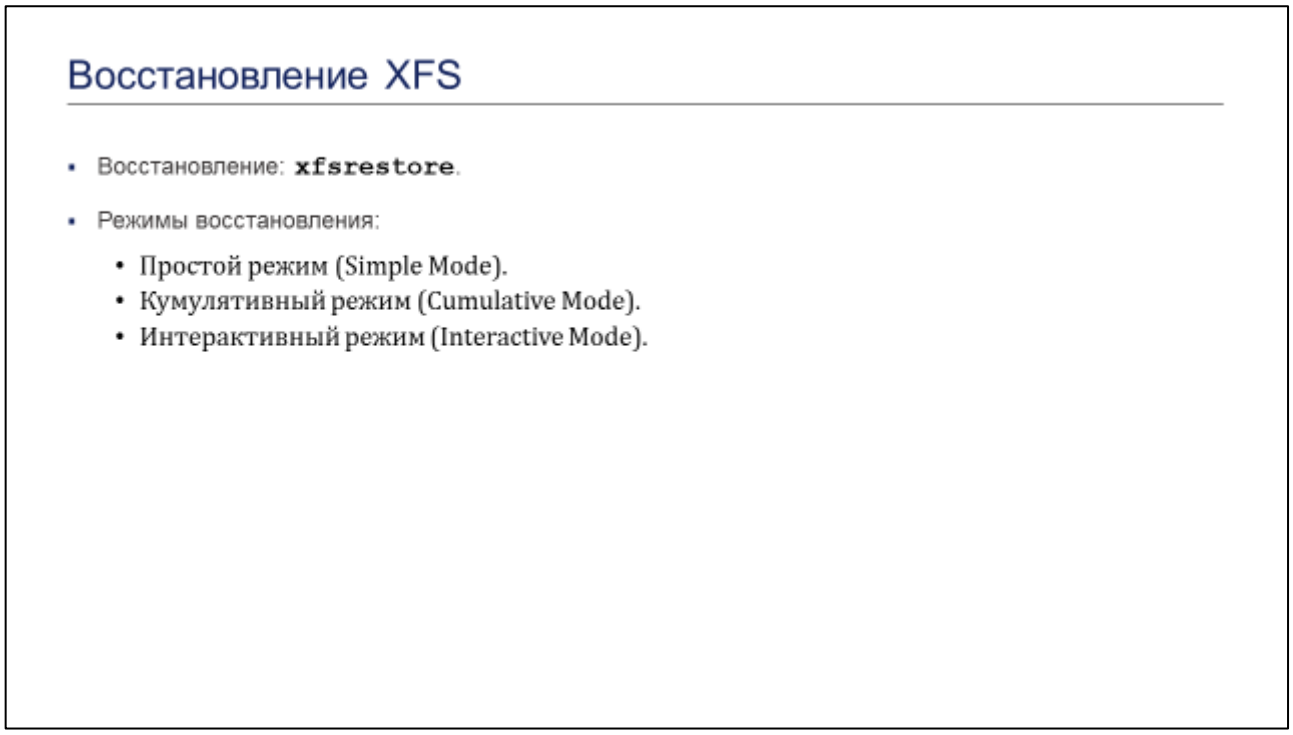
```
# xfsdump -l 0 -f /backup-files/boot.xfsdump /boot
# xfsdump -l 0 -f /backup-files/data.xfsdump /data
```

Для резервного копирования нескольких файловых систем на одно ленточное устройство **/dev/st0**:

```
# xfsdump -l 0 -L "backup_boot" -f /dev/st0 /boot
# xfsdump -l 0 -L "backup_data" -f /dev/st0 /data
```

Дополнительные ресурсы:

- Страница руководства **xfsdump(8)**.



Системный администратор может использовать утилиту **xfsrestore** для восстановления файловой системы XFS из резервной копии, сделанной при помощи **xfsdump**.

Возможности восстановления XFS из резервной копии.

Утилита **xfsrestore** восстанавливает файловые системы из резервных копий, сделанных при помощи xfsdump. Утилита **xfsrestore** имеет два режима:

- **Простой режим (Simple Mode)** – он позволяет пользователям восстанавливать всю файловую систему из дампа с уровнем **0**. Это режим по умолчанию.
- **Кумулятивный режим (Cumulative Mode)** – он позволяет восстанавливаться из инкрементальных резервных копий, который имеют уровни дампа от **1** до **9**.

Уникальный идентификатор сессии (Session ID) или заголовок сессии (Session Label) определяют каждую резервную копию. Восстановление резервной копии с ленты, содержащей несколько резервных копий, требуют указания соответствующего идентификатора или заголовка.

Для извлечения, добавления или удаления определенных файлов из резервной копии необходимо воспользоваться интерактивным режимом (Interactive Mode) **xfsrestore**.

Дополнительные ресурсы:

- Страница руководства **xfsrestore(8)**.

Восстановление файловой системы из резервной копии.

Системный администратор может восстанавливать содержимое файловой системы XFS из резервной копии в файле или на ленте.

Предварительные требования:

- Резервная копия файловой системы в файле или на ленте.
- Устройство хранения для восстановления файловой системы XFS.

Процедура:

Команда восстановления резервной копии варьируется в зависимости от того где хранится резервная копия и какой уровень дампа у резервной копии:

```
# xfsrestore [-r] [-S session-id] [-L session-label] [-i] \
-f backup-location restoration-path
```

- Замените *backup-location* на расположение резервной копии. Это может быть обычный файл, ленточный накопитель, удаленное ленточное устройство. Например, файл **/backup-files/Data.xfsdump** или ленточный накопитель **/dev/st0**.
- Замените *restoration-path* на путь к каталогум, где необходимо восстановить файловую систему. Например, **/mnt/data**.

Для восстановления файловой системы из инкрементальной резервной копии (уровни с **1** по **9**), необходимо добавить опцию **-r**.

Для восстановления резервной копии с ленточного накопителя, содержащего несколько резервных копий, необходимо указать опции **-S** или **-L**. Опция **-S** позволяет выбрать резервную копию по идентификатору сессии (Session ID), а опция **-L** – по заголовку сессии (Session Label). Для получения идентификатора сессии и заголовка сессии, воспользуйтесь командой **xfsrestore -l**.

- Замените *session-id* идентификатором сессии резервной копии. Например, **b74a3586-e52e-4a4a-8775-c3334fa8ea2c**. Замените *session-label* заголовком сессии резервной копии. Например, **my_backup_session_label**.

Чтобы использовать утилиту **xfsrestore** интерактивно необходимо воспользоваться опцией **-i**.

Интерактивный диалог начинает после того, как **xfsrestore** прочитает указанное устройство. Доступные команды в интерактивной оболочке xfsrestore включают **cd**, **ls**, **add**, **delete** и **extract**. Для получения полного списка команд можно воспользоваться командой help.

Пример восстановления нескольких файловых систем XFS.

Восстановление из файлов резервных копий и сохранения их содержимого в каталогах внутри **/mnt**:

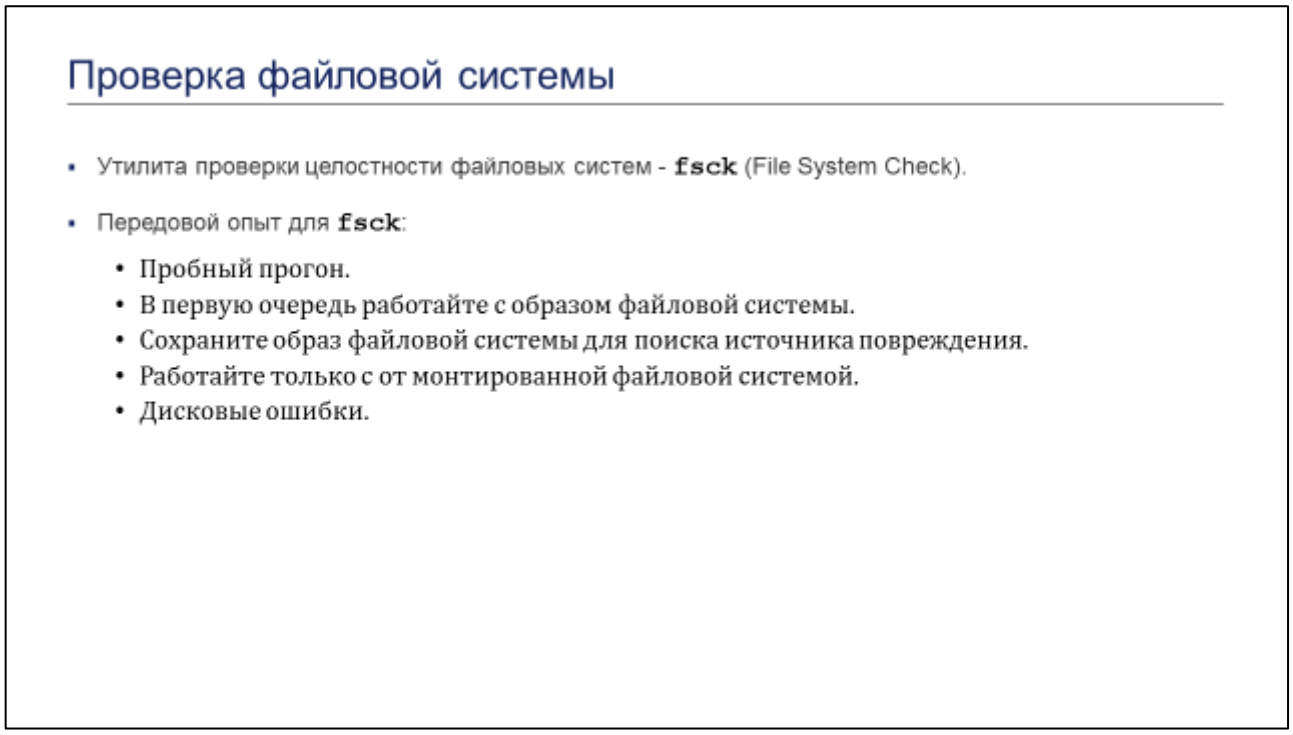
```
# xfsrestore -f /backup-files/boot.xfsdump /mnt/boot/
# xfsrestore -f /backup-files/data.xfsdump /mnt/data/
```

Восстановление с ленточного устройства, содержащего несколько резервных копий, укажите каждую резерную копию по ее заголовку сессии (Session Label) или идентификатору сессии (Session ID).

```
# xfsrestore -L "backup_boot" -f /dev/st0 /mnt/boot/
# xfsrestore -S "45e9af35-efd2-4244-87bc-4762e476cbab" -f /dev/st0 /mnt/data/
```

Дополнительные ресурсы:

- Страница руководства **`xfsrestore(8)`**.



Файловая система может быть проверена на целостность и опционально исправлена, при помощи специфичных утилит для файловой системы. Как правило эти утилиты имеют отношение к **`fsck`** (сокращение от File System Check).

ПРИМЕЧАНИЕ

Утилиты для проверки файловых систем, гарантируют только целостность метаданных, они не умеют работать с реальными данными, содержащимися на файловой системе и не являются инструментами восстановления данных.

Нарушение целостности файловой системы может случить по различным причинам, в том числе из-за аппаратных ошибок, ошибок администратора хранилища и программных неисправностей.

До того, как современные файловые системы с ведением журнала метаданных стали популярными, каждый сбой электропитания и некорректная перезагрузка требовали проверки файловой системы. Это происходило из-за того, что каждое прерванное обновление файловой системы, приводило ее в состояние нарушения целостности. В результате, проверка каждой файловой системы, указанной в **`/etc/fstab`** автоматически запускалась во время загрузки. Для файловых систем с журналированием метаданных, эта операция проходит очень быстро, так как при помощи журнала метаданных проверить целостность значительно проще, даже после аварии.

Однако, возможны ситуации нарушения целостности файловой системы и повреждения данных, даже для журналируемых файловых систем. В таких случаях необходимо использовать проверщик для починки файловой системы.

Передовой опыт для fsck. Обычно, при запуске утилиты проверки и починки файловой системы, в случае нахождения нарушений целостности, происходит автоматическая починка файловой системы. В некоторых случаях, серьезно поврежденные иноды (inodes) или каталогу могут быть

удалены, если их невозможно починить. Чтобы убедить что неожиданные и нежелательные изменения не произойдут, воспользуйтесь следующими шагами:

Пробный прогон. Большинство проверщиков файловых систем поддерживают режим, который проверяет, но не исправляет файловую систему. В этом режиме проверщик выведет любые найденные ошибки и действия, которые он может предпринять, не внося никаких изменений на файловую систему.

В первую очередь работайте с образом файловой системы. Большинство файловых систем поддерживают создание образа метаданных (Metadata Image) - выборочной копии файловой системы, которая содержит только метаданные. Так как проверщик файловой системы работает только с метаданными, можно выполнить пробный прогон на образе метаданных, чтобы оценить какие изменения будут внесены. Если изменения будут приемлемыми, то затем можно выполнить операцию почини на файловой системе.

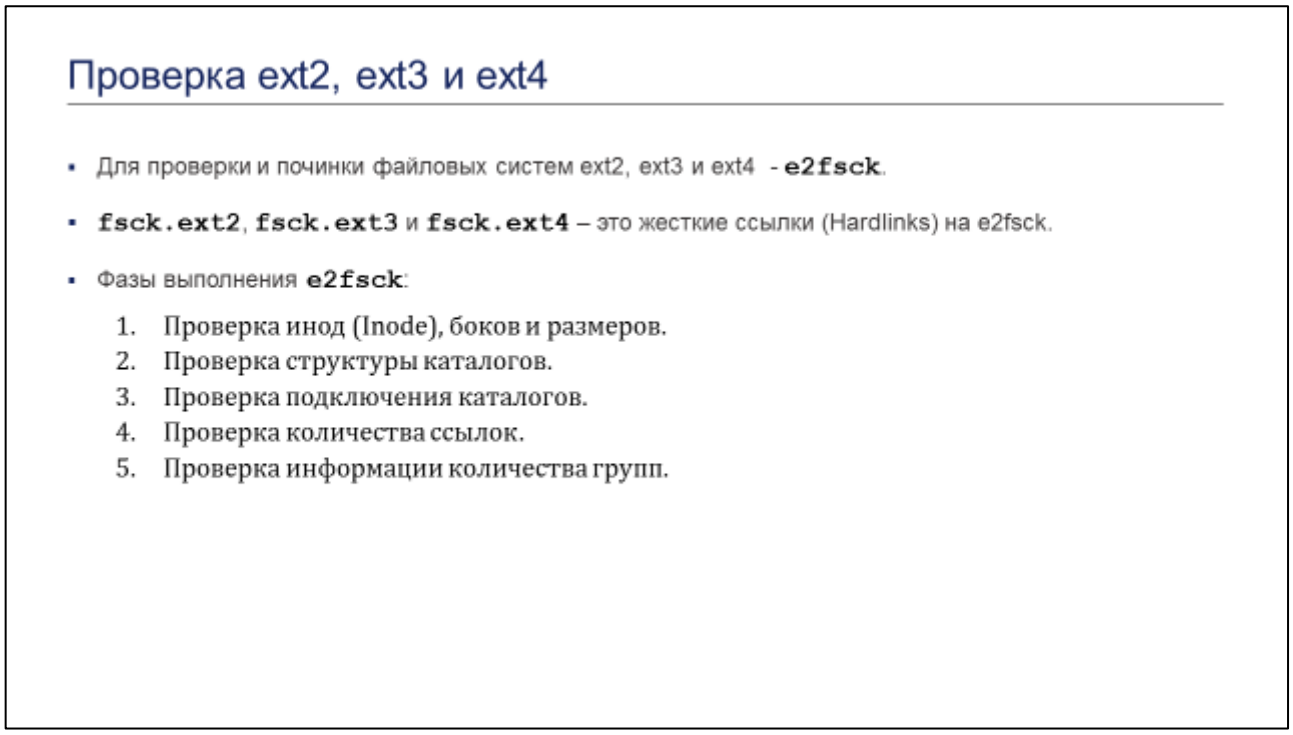
ПРИМЕЧАНИЕ

Серьезно поврежденная файловая система может вызвать проблему при создании образа метаданных.

Сохраните образ файловой системы для поиска источника повреждения. Образ мета данных, созданный до операции починки файловой системы, может быть полезен при поиске причин повреждения файловой системы, вызванной ошибкой программного обеспечения.

Работайте только с от монтированной файловой системой. Починка файловой системы может быть выполнена только на отмонтированной файловой системе. Инструмент починки должен иметь индивидуальный однопользовательский доступ к файловой системе, чтобы избежать дальнейшего повреждения файловой системы. Некоторые инструменты позволяют выполнять проверку без починки на смонтированной файловой системе, такая проверка может показать не существующие ошибки, если файлы будут заняты сторонними процессами.

Дисковые ошибки. Инструменты проверки файловой системы не могут исправить аппаратные проблемы. Файловая система должна быть полностью доступна для чтения и записи, чтобы операция починки завершилась успешно. Если файловая система повреждена в следствии аппаратных ошибок, ее можно переместить на другой диск при помощи утилиты **dd(8)**.



Файловые системы ext2, ext3 и ext4 используют исполняемый файл **e2fsck** для проверки и починки файловой системы. Имена файлов **fsck.ext2**, **fsck.ext3** и **fsck.ext4** – это жесткие ссылки (Hardlinks) на исполняемый файл **e2fsck**. Эти исполняемые файлы запускаются автоматически во время загрузки и их поведение зависит от состояния файловой системы и момента запуска проверки.

Полная проверка вызывается на файловой системе ext2, которая не поддерживает ведение журнала метаданных файловой системы и для ext4, если отключено журналирование.

Для файловых систем ext3 и ext4 с журналированием метаданных, журнал повторно выполняет в пространстве пользователя и исполняемый файл завершает свою работу. Это является действием по умолчанию, так как повторной выполнение журнала обеспечивает целостность файловой системы после аварии.

Если обнаруживается нарушение целостности файловой системы, когда она смонтирована, эта информация записывается в суперблок. Если **e2fsck** находит эту отметку в суперблоке, то она выполняет полную проверку после повторного применения журнала (если журнал ведется).

e2fsck может запросить пользовательский ввод в процессе выполнения, если не указана опция -p. Опция -p сообщает **e2fsck** автоматически выполнять все исправления, если они могут выполниться безопасно. Если требуется участие пользователя **e2fsck** отображает этот статус в выходном коде.

ПРИМЕЧАНИЕ

Выходные коды (Exit Code) возвращаемые **e2fsck**, это суммы следующих статусов:

- **0** – ошибки отсутствуют.
- **1** – ошибки файловой системы исправлены.
- **2** – ошибки файловой системы исправлены, система должна быть перезагружена.
- **4** – ошибки файловой системы остались неисправленными.
- **8** – ошибка выполнения.
- **16** – ошибка использования или синтаксиса.
- **32** – **e2fsck** была отменена пользователем.
- **128** – ошибка общей библиотеки.

Популярные опции e2fsck:

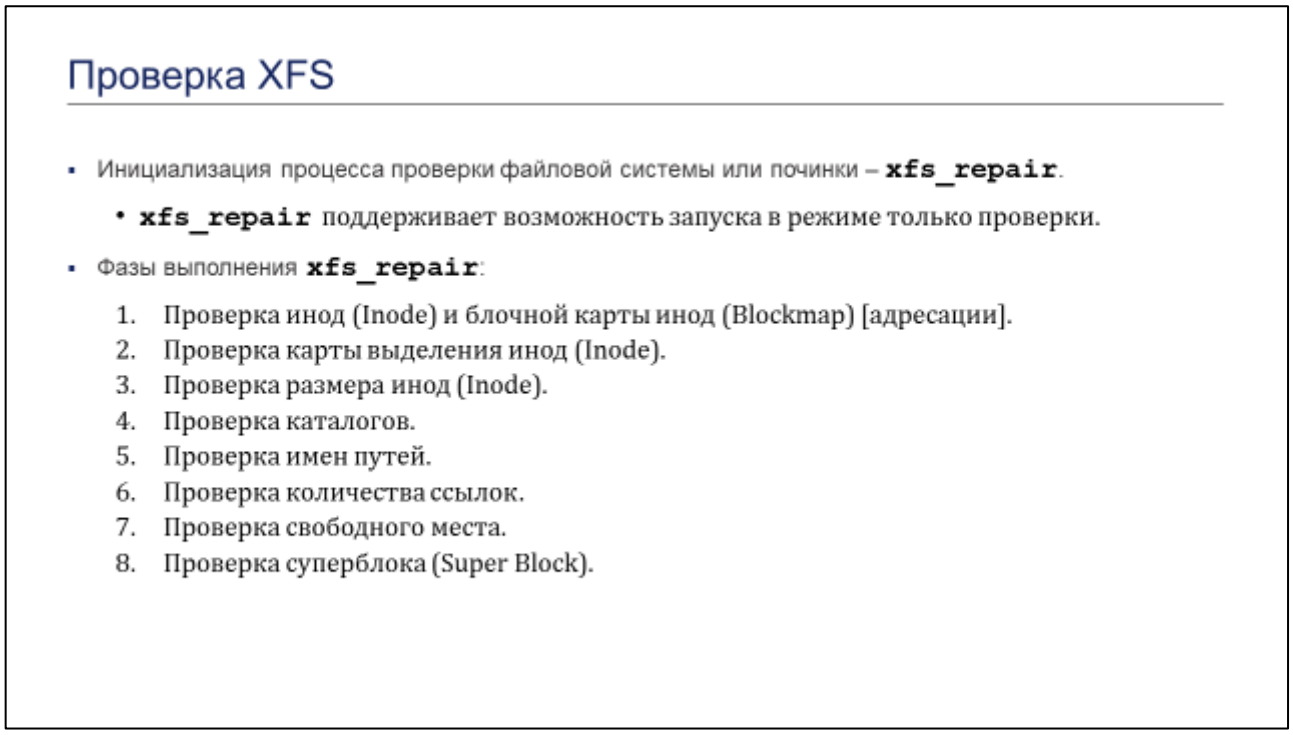
Опция	Опимание
-n	Режим без изменений, только проверка.
-b суперблок	Указывает номер блока, в качестве альтернативы суперблоку, если первый поврежден.
-f	Принудительный запуск полной проверки, даже если в суперблоке не содержится информации об ошибках.
-j журнал	Указывает путь к журналу.
-p	Автоматически чинит или «приводит в порядок» файловую систему, без необходимости пользовательского ввода.
-y	Принимает ответ «да» на все выдаваемые вопросы.

Дополнительную информацию по всем опциям можно найти на странице руководства **e2fsck(8)**.

Следующие пять фаз выполняются, когда запущена e2fsck:

1. Проверка инод, боков и размеров.
2. Проверка структуры каталогов.
3. Проверка подключения каталогов.
4. Проверка количества ссылок.
5. Проверка информации количества групп.

Утилита **e2image(8)** может быть использована для создания образа метаданных до операции починки или в целях тестирования и диагностики. Опция **-r** должна быть использована в целях тестирования для создания выборочного файла размером с файловую систему. **e2fsck** может напрямую работать с результирующим файлом. Опция **-Q** должна быть использована для указания на образ, который должен быть заархивирован или использован для диагностики. Это создает более компактный формат файла, подходящий для передачи.



Никакие операции восстановления не выполняются автоматически в процессе загрузки. Для инициализации процесса проверки файловой системы или починки, должна быть использована утилита **xfs_repair**.

ПРИМЕЧАНИЕ

В пакете xfsprogs содержится утилита **fsck.xfs**, она предоставлена только для соответствия инициализирующим скрипатом, которые выглядят следующим образом **fsck.***фс*. Утилита **fsck.xfs** немедленно завершает свою работу с выходным кодом **0**.

Более старая версия пакета **xfsprogs** содержит утилиту **xfs_check**. Это очень медленная утилита, не поддерживающая большие файловые системы. В результате она устарела и более не используется, так как появилась возможность запускать **xfs_repair -n**.

Чистый журнал на файловой системе требуется для работы **xfs_reapir**. Если файловая система не была корректно отмонтирована, она должна быть смонтирована и отмонтированна заново перед использованием **xfs_repair**. Если журнал поврежден и не может быть воспроизведен, можно использовать опцию **-L** для обнуления журнала.

ПРИМЕЧАНИЕ

Опция **-L** должна быть использована только в том случае, когда журнал не может быть воспроизведен. Опция отменяет все обновления метаданных в журнале, что может привести к нарушению. Целостности в будущем.

xfs_reapir поддерживает возможность запуска в режиме только проверки при помощи опции **-n**. В этом случае никакие изменения не будут внесены на файловую систему.

Наиболее часто используемые опции xfs_repair:

- **-n** – Режим проверки без изменений.
- **-L** – Обнуление журнала метаданных.
- **-m** *maxmem* – Ограничение памяти во время запуска до значения maxmem MB. **0** может быть указан для запуска с минимально возможным объемом памяти.
- **-l** *logdev* – Указывает внешнее устройство для ведения журнала.

Подробное описание опций и их использования представлены на странице руководства **xfs_reapir(8)**.

Следующие 8 основных фаз выполняет **xfs_repair** после запуска:

1. Проверка инод (Inode) и блочной карты инод (Blockmap) [адресации].
2. Проверка карты выделения инод (Inode).
3. Проверка размера инод (Inode).
4. Проверка каталогов.
5. Проверка имен путей.
6. Проверка количества ссылок.
7. Проверка свободного места.
8. Проверка суперблока (Super Block).

Эти фазы и сообщения, которые выводятся в процессе работы подробно описаны на странице руководства **xfs_reapir(8)**.

xfs_reapir не интерактивна. Все операции выполняются автоматически без участия пользователя.

Рекомендуется создать образ метаданных до начала починки для последующей диагностики и тестирования, для этого можно использовать утилиты **xfs_metadump(8)** и **xfs_mdrestore(8)**.

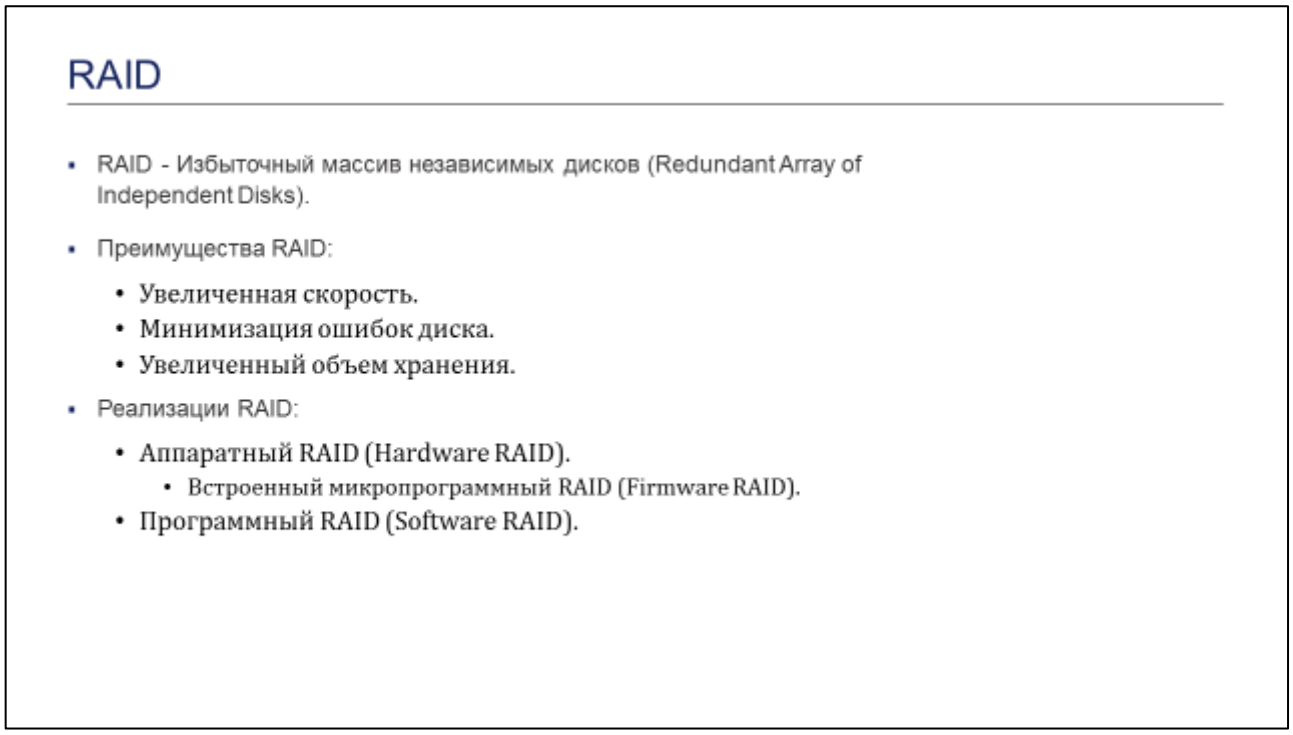
Модуль 3. RAID и LVM в РЕД ОС

В этом модуле рассматриваются:

Типы логических накопителей

Программные RAID-массивы

Менеджер логических томов (LVM)



Избыточный массив независимых дисков (Redundant Array of Independent Disks, RAID). Основной идеей, находящейся в основе RAID, является объединение нескольких, недорогих дисковых накопителей в массив для достижения производительности или отказоустойчивости, что недостижимо при помощи одного большого и дорогого накопителя.

RAID позволяет получить доступ к нескольким дискам. RAID использует такие технологии, как чередование дисков (Disk Striping, RAID0), зеркалирование дисков (Disk Mirroring, RAID1) и чередование дисков с контролем четности (Disk Striping with Parity, RAID5) для достижения отказоустойчивости, меньшей задержки, увеличения пропускной способности и максимизации восстановления данных при выходе дисков из строя.

RAID целостно распространяет данные между каждым диском в массиве. RAID разбивает данные на равные части (Обычно 32Kb или 64 Kb, хотя можно использовать и другие размеры). Каждая часть записывается на диск в RAID массиве в соответствии с выбранным уровнем RAID. Когда данные считываются, процесс обращается, давая представление, что множеств дисков в массиве являются одним большим накопителем.

Преимущества от использования RAID.

Системные администраторы и прочие, кто управляет большими объемами данных получат преимущества от использования технологии RAID. Основные преимущества от использования RAID:

- Увеличенная скорость.
- Увеличенный объем хранения при помощи одного виртуального диска.
- Минимизация ошибок диска.

Типы RAID.

Существует три подхода к реализации RAID: встроенный микропрограммный RAID (Firmware RAID), аппаратный RAID (Hardware RAID) и программный RAID (Software RAID).

Встроенный микропрограммный RAID (также известный как ATARAID) – это тип программного RAID, при котором набор RAID настраивается при помощи микропрограммного обеспечение, которое обычно встроено в BIOS, что позволяет грузиться с этого RAID массива. Различные производители используют различные мета-данные диска для маркировки членов RAID-набора. Intel Matrix RAID – это хороший пример системы встроенного микропрограммного RAID.

Аппаратный RAID (Hardware RAID) – массив на основе аппаратного RAID контроллера управляется независимо от хоста. Это предоставляет один диск на RAID массив хосту.

Аппаратные RAID устройства подключаются к SCSI контроллеру и предоставляют RAID массив в качестве одиночного SCSI накопителя. Внешняя RAID система перемещает всю обработку на отдельный контроллер, расположенный в отдельной дисковой подсистеме. Вся подсистема подключается к хосту через обычный SCSI контроллер и представляется хосту как один диск.

Карта RAID-контроллера функционирует как SCSI контроллер в операционной системе и обрабатывает все текущие дисковые коммуникации. К RAID контроллеру подключаются накопители (так же как к обычному SCSI-контроллеру) и затем добавляются в конфигурацию RAID контроллера и предоставляются операционной системе, которая не видит разницы между SCSI-контроллером и RAID-контроллером.

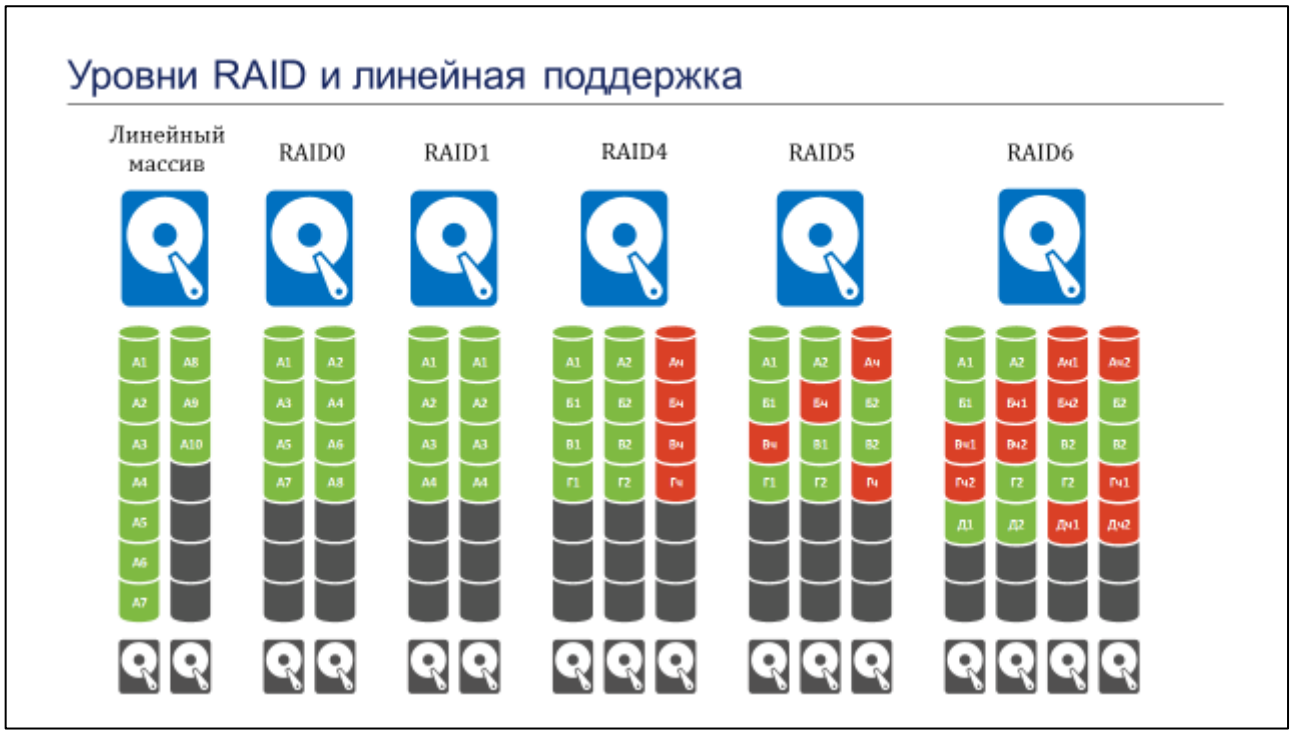
Программный RAID (Software RAID) применяет различные уровни RAID в коде ядра диска (блочного устройства). Это предоставляет максимально дешёвое решением по сравнению с аппаратным, если не требуется горячая замена. Программный RAID работает и с дешёвыми дисками (IDE) и со SCSI-дисками. С современными быстрыми процессорами, программный RAID имеет преимущество над аппаратным RAID.

Ядро Linux содержит драйвер multi-disk (MD), который позволяет решению RAID быть полностью аппаратно-независимым. Производительность программного массива зависит от производительности и загруженности процессора.

Основные возможности программного RAID в Linux:

- Многопоточная структура.
- Возможность перемещение массивов между Linux-машинами без перенастройки.
- Фоновая реконструкция при помощи простаивающих системных ресурсов.
- Поддержка подключения накопителей на горячую.
- Автоматическое определение CPU для получения преимуществ от определенной оптимизации CPU.
- Регулярные проверки целостности данный RAID, для проверки здоровья массива.
- Интерактивный мониторинг массива с возможностью уведомления по почте о важных событиях.
- Битовые карты записи, которые увеличивают скорость событий ресинхронизации, указывая ядру какие части массива должны быть синхронизированы, вместо синхронизации всего массива.
- Моментальные снимки ресинхронизации, замораживающие синхронизацию на время перезагрузки системы.

- Возможность изменения массива после создания, например, расширения массива без остановки.



RAID поддерживает различные конфигурации, в том числе уровни 0, 1, 4, 5, 6, 10 и линейный.

Уровень 0.

Уровень RAID 0, также называемый чередование (Striping) –это ориентированная на производительность техника чередования данных. Эта техника позволяет разбивать данные на части и записывать их на все диски массива, предоставляя высокую производительность ввода/вывода, низкую стоимость и не предоставляя отказоустойчивости. Объем хранилища в массиве с уровнем 0 равен сумме объемов дисков в аппаратном RAID или сумме разделов в программном RAID.

Уровень 1.

Уровень RAID 1 или зеркалирование используется дольше, чем любой другой уровень RAID. Уровень 1 предоставляет отказоустойчивость за счет записи данных на каждый диск в массиве, оставляя зеркальную копию на каждом диске. Зеркалирование остается популярным за счет простоты и высокого уровня доступности данных. Уровень 1 оперирует двумя или более дисками, что позволяет использовать параллельный доступ при чтении, но как правило оперируют независимо для предоставления высоких показателей ввода/вывода. Уровень 1 предоставляет очень хорошую надежность и увеличивает производительность приложений с высокими потребностями в чтении данных, при этом с достаточно высокой стоимостью. Объем массива с уровнем 1 равен объему одного диска в аппаратном RAID или одного массива в программном RAID.

ПРИМЕЧАНИЕ

Для записи 10 Gb данных в RAID1 состоящем из двух дисков потребуется записать 20 Gb, по 10 Gb на каждый из накопителей.

Уровень 4.

Уровень RAID 4 использует четность, содержащуюся на одном диске для защиты данных. Это лучше подходит для операций ввода/вывода а не передачи больших файлов. Так как выделенный для четности диск выступает узким местом решения, поэтому уровень 4 редко используется без сопутствующих технологий, таких как кэширование. Также RAID уровня 4 не поддерживается в CentOS. Объем дискового пространства в аппаратных RAID уровня 4 равен сумме объемов дисков минус объем одного диска, выделенного для четности.

ПРИМЕЧАНИЕ

Объем массива RAID4 равен объему массива RAID5, но уровень 5 имеет ряд преимуществ, поэтому уровень 4 не рекомендуется использовать в RedOS.

Уровень 5.

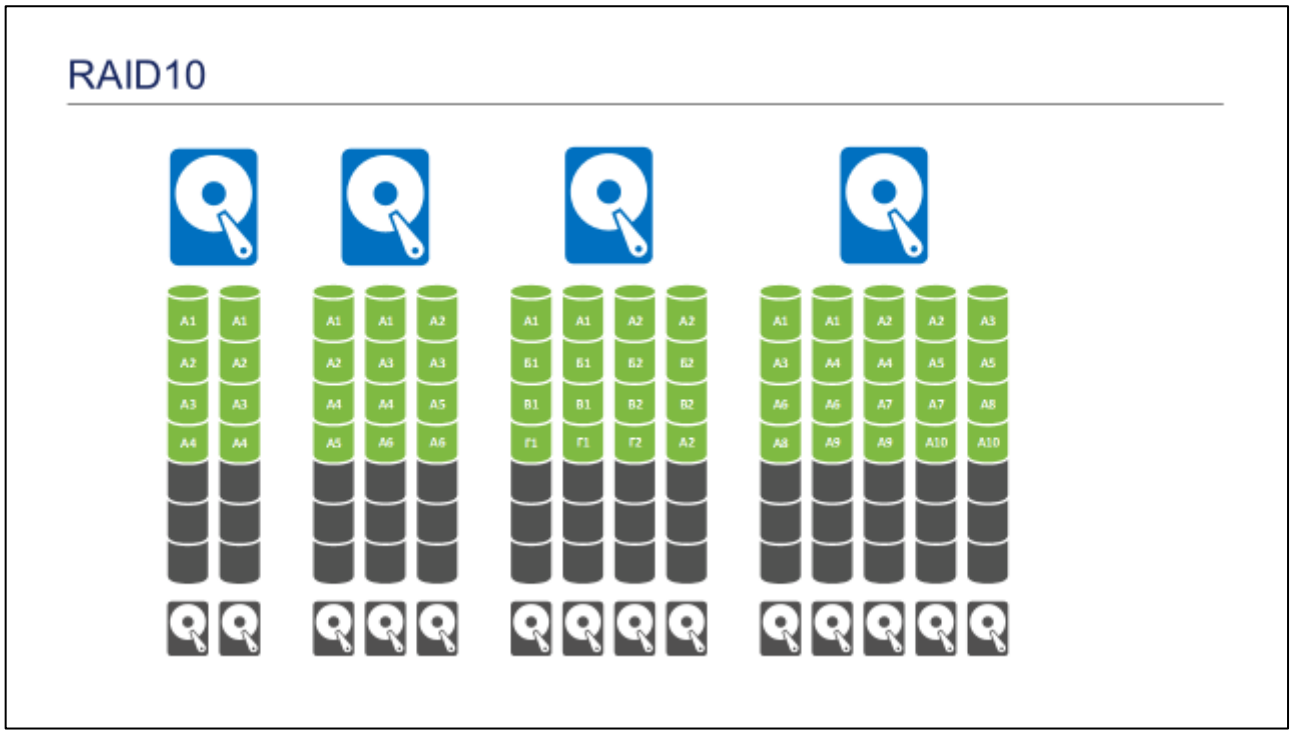
Уровень RAID 5 наиболее популярный тип RAID. За счет распространения четности между несколькими или всеми дисками в массиве, уровень 5 не наследует узкое место при записи от уровня 4. Узким местом остается только вычисление четности, что не является проблемой для программных массивов RAID с современными процессорами. Также, как и в уровне 4 чтение значительно снижает скорость записи. Уровень 5 часто используется вместе с кэшированием, что существенно снижает асимметрию в производительности. Объем дискового пространства в аппаратных RAID уровня 5 равен сумме объемов дисков минус объем одного диска, а для программных равен сумме объемов разделов минус объем одного раздела.

Уровень 6.

Уровень 6 – типичный уровень RAID, при котором данные избыточны и сохранны, но менее производительны. RAID 6 использует комплексную схему четности, которая позволяет восстановиться при потере любых двух дисков в наборе. Эта комплексная схема четности создает гораздо больше нагрузок на процессоре в процессе записи, что существенно увеличивает асимметрию при записи по сравнению с уровнями 4 и 5.

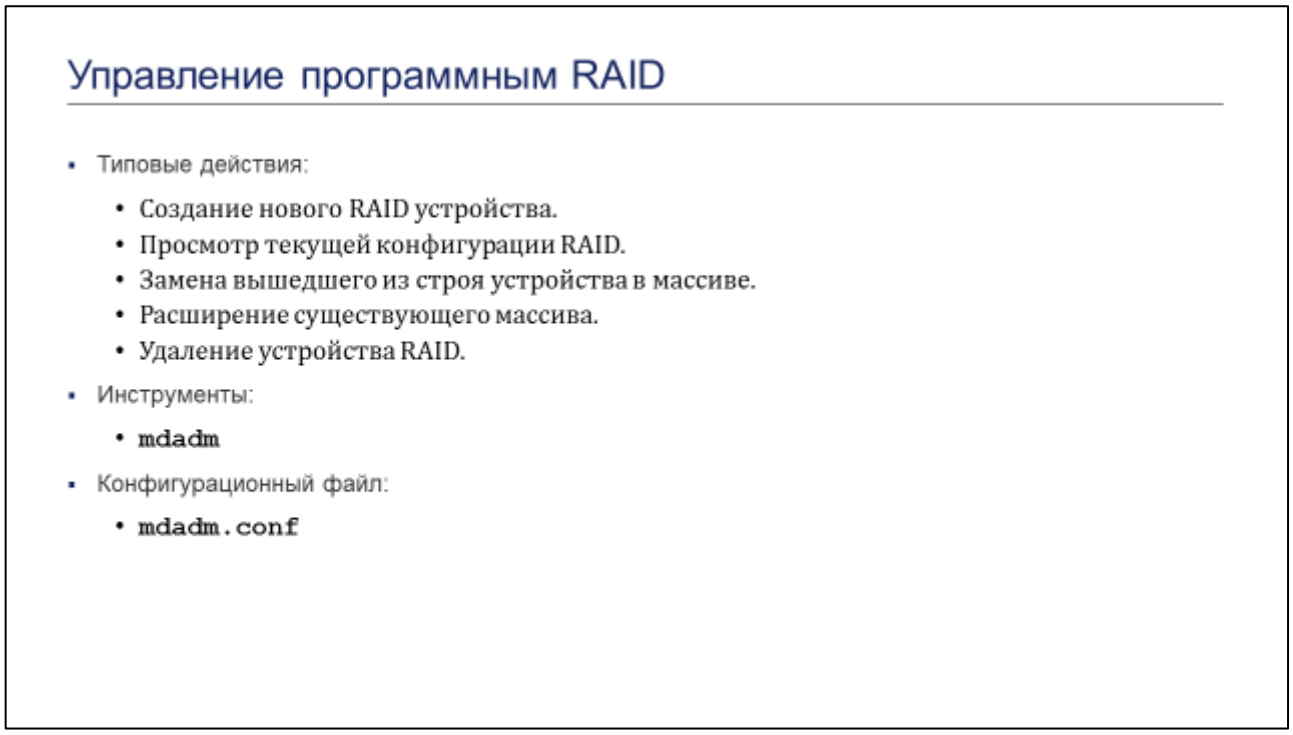
Линейный RAID.

Линейный RAID – это простая группировка накопителей для создания одного большого виртуального накопителя. В линейном RAID части выделяются последовательно, заполняя накопители один за другим, переход к заполнению следующего накопителя происходит после полного заполнения предыдущего. Такая группировка не предоставляет никаких преимуществ, кроме одного большого виртуального накопителя, также снижается стабильность из-за потери одного накопителя весь массив будет непригоден. Объем линейного RAID равен сумме объемов дисков в массиве.



RAID 10.

Уровень 10 позволяет комбинировать преимущества производительности уровня 0 и избыточности уровня 1. Также этот уровень позволяет сократить накладные расходы объема диска, присущие RAID1. При помощи уровня 10 возможно создать массив из 3 дисков, где данные будут храниться только дважды, что позволит создать массив размером в полтора раза больше самого маленького устройства в массиве.



Для создания нового RAID устройства, выполните следующую команду, от имени пользователя root:

```
# mdadm --create raid_device --level=level --raid-devices=number device…
```

Это самый простой путь для создания RAID массива. В команде присутствует множество опций, которые позволяют указывать число подключаемых устройств, размер блока чередующегося массива и так далее. Все эти опции будут иметь влияние на производительность, для получения детальной информации по ним ознакомьтесь с секцией CREATE MODE на странице руководства **mdadm(8)**.

Пример создания нового RAID устройства:

Представим, что у нас есть два неиспользованных SCSI-диска и на каждом из них есть раздел одинакового размера.

```
# ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sdb  /dev/sdb1  /dev/sdc  /dev/sdc1  /dev/sdd
```

Для создания **/dev/md0**, как нового RAID1 массива из **/dev/sdb1** и **/dev/sdc1**, запустите следующую команду:

```
# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Просмотр текущей конфигурации RAID.

Когда используется программный RAID, базовая информация обо всех активных RAID-устройствах записывается в специальном файле **/proc/mdstat**. Для просмотра файла можно воспользоваться командой cat:

```
# cat /proc/mdstat
```

Для определения, когда устройство является RAID устройством, а когда компонентным, выполните следующую команду от имени пользователя root:

```
# mdadm --query device…
```

Для получения детальной информации о RAID-устройстве, используйте следующую команду:

```
# mdadm --detail raid_device…
```

Аналогично, проверки компонентного устройства, введите:

```
# mdadm --examine component_device…
```

Команда **mdadm --detail** отображает информацию о RAID-устройстве, **mdadm --examine** только информацию, относящуюся к компонентному устройству. Это различие имеет значение, когда RAID устройство является компонентом другого RAID-устройства.

Команда **mdadm --query**, также, как и команды **mdadm --detail** и **mdadm --examine** позволяют вам указывать несколько устройств за раз.

Пример просмотра конфигурации RAID.

Проверки, что устройство **/dev/md0** является RAID устройством:

```
# mdadm --query /dev/md0
/dev/md0: 99.94GiB raid1 2 devices, 0 spares. Use mdadm --detail for more detail.
```

Как видно из вывода, эта команда предоставляет краткий обзор RAID-устройства и его конфигурации. Для отображения подробной информации воспользуемся следующей командой:

```
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Jan  2 01:27:29 2016
     Raid Level : raid1
     Array Size : 104791040 (99.94 GiB 107.31 GB)
  Used Dev Size : 104791040 (99.94 GiB 107.31 GB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Sat Jan  2 01:36:11 2016
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : c7-server01.nerv.local:0  (local to host c7-server01.nerv.local)
           UUID : 8119886b:63181d89:9d29e86f:f2ab9565
         Events : 17
```

```
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       1       8       33        1      active sync   /dev/sdc1
```

Для вывода всех активных RAID устройств, введите:

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdc1[1] sdb1[0]
      104791040 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

Замена вышедшего из строя устройства в массиве.

Для замены необходимого устройства в программном RAID, в первую очередь убедитесь, что устройство помечено как неисправное (Faulty), при помощи следующей команды:

```
# mdadm raid_device --fail component_device
```

Затем удалите неисправное устройство:

```
# mdadm raid_device --remove component_device
```

После починки устройства, вы можете повторно добавить его в массив:

```
# mdadm raid_device --add component_device
```

Пример замены неисправного устройства:

У нас в системе присутствует следующее RAID-устройство **/dev/md0**:

```
# mdadm --detail /dev/md0 | tail -n 3
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       1       8       33        1      active sync   /dev/sdc1
```

Предположим первый диск вышел из строя и должен быть заменен, для этого пометим **/dev/sdc1** как неисправный:

```
# mdadm /dev/md0 --fail /dev/sdc1
mdadm: set /dev/sdc1 faulty in /dev/md0
```

Затем удалим его из RAID-устройства:

```
# mdadm /dev/md0 --remove /dev/sdc1
mdadm: hot removed /dev/sdc1 from /dev/md0
```

После замены оборудования, можно вернуть диск /dev/sdc1 на место в массив:

```
# mdadm /dev/md0 --add /dev/sdc1
mdadm: added /dev/sdc1
```

Расширение существующего массива.

Для добавления нового устройства в существующий массив, используйте следующую команду, с правами пользователя root:

```
# mdadm raid_device --add component_device
```

Это добавит устройство в качестве запасного устройства. Для расширения массива и активного использования устройства введите следующую команду:

```
# mdadm --grow raid_device --raid-devices=number
```

Пример расширения RAID-устройства.

Предположим в системе имеется активное RAID-устройство, **/dev/md0**:

```
# mdadm --detail /dev/md0 | tail -n 3
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       2       8       33        1      active sync   /dev/sdc1
```

Также имеется дисковый накопитель SCSI **/dev/sdd** с одним разделом. Для добавления раздела в массив **/dev/md0**, воспользуемся следующей командой:

```
# mdadm /dev/md0 --add /dev/sdd1
mdadm: added /dev/sdd1
# mdadm --detail /dev/md0 | tail -n 5
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       2       8       33        1      active sync   /dev/sdc1

       3       8       49        -      spare   /dev/sdd1
```

Это добавить запасное устройство, для того чтобы задействовать устройство в массиве, необходимо использовать следующую команду:

```
# mdadm --grow /dev/md0 --raid-devices=3
raid_disks for /dev/md0 set to 3
unfreeze
# mdadm --detail /dev/md0 | tail -n 4
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       2       8       33        1      active sync   /dev/sdc1
       3       8       49        2      spare rebuilding   /dev/sdd1
```

Удаление устройства RAID.

Для удаления RAID-устройства, в первую очередь необходим его деактивировать:

```
# mdadm --stop raid_device
```

После деактивации можно удалить RAID-устройство:

```
# mdadm --remove raid_device
```

В конце нужно обнулить супеблок всех устройств, которые участвовали в массиве:

```
# mdadm --zero -superblock component_device…
```

Пример удаления RAID-устройства.

Предположим в системе имеется активное RAID-устройство, **/dev/md0**:

```
# mdadm --detail /dev/md0 | tail -n 4
```

```
    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       2       8       33        1      active sync   /dev/sdc1
       3       8       49        2      active sync   /dev/sdd1
```

Для удаления устройства, в первую очередь деактивируем его:

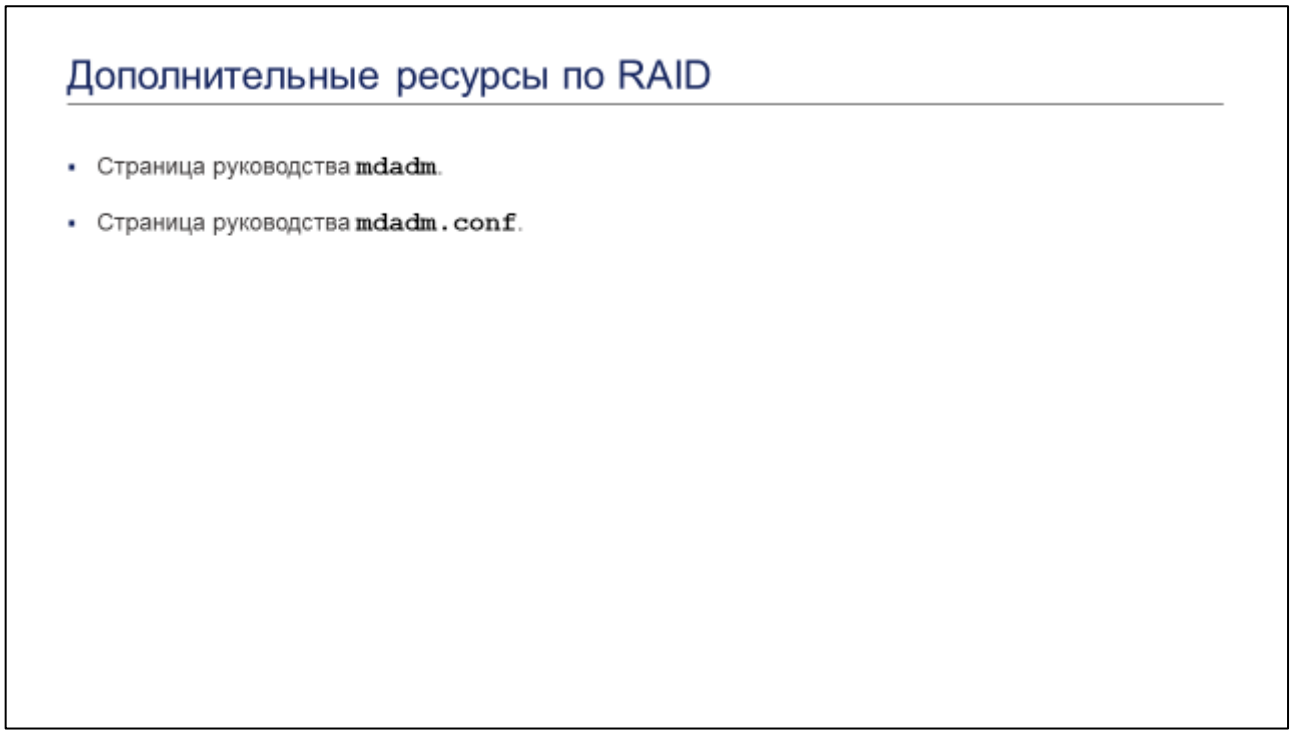
```
# mdadm --stop /dev/md0
mdadm: stopped /dev/md0
```

После остановки устройства удалим его (Если вы не делали сохранение конфигурации, этот шаг не потребуется):

```
# mdadm --remove /dev/md0
mdadm: error opening /dev/md0: No such file or directory
```

Для завершения процесса удаления устройства, обнулим суперблок всех компонентных устройств:

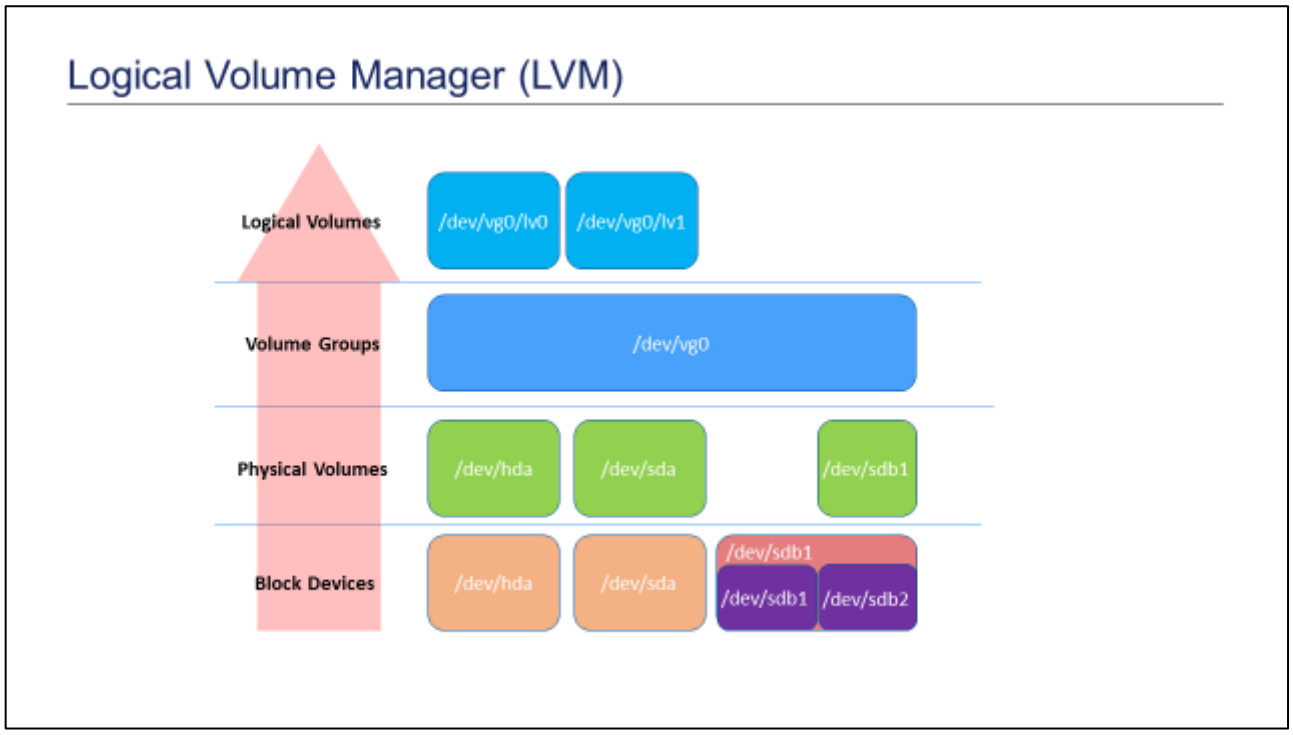
```
# mdadm --zero-superblock /dev/sdb1 /dev/sdc1 /dev/sdd1
```



За дополнительной информацией о RAID обратитесь к следующим ресурсам.

Установленная документация:

- Страница руководства **mdadm** – описание утилиты **mdadm**.
- Страница руководства **mdadm.conf** – описание полного списка конфигурационных опций, доступных в **/etc/mdadm.conf**.



LVM (Logical Volume Manager), менеджер логических томов – это система управления дисковым пространством, абстрагирующаяся от физических устройств. Эта система позволяет легко управлять дисковым пространством, добавлять тома, изменять их размер и т.д.

Менеджер логических томов оперирует несколькими основными понятиями:

- **PV (Physical Volume)** – это раздел диска, либо раздел RAID-массива. Физический том входит в состав группы томов;
- **VG (Volume Group)** – группа, объединяющая несколько физических томов, является верхним уровнем абстрактной модели LVM;
- **LV (Logical Volume)** – раздел группы томов. Для более простого понимания, можно представить группу томов в качестве жёсткого диска, а логические тома в качестве разделов этого диска;
- **PE (Physical Extent)** – каждый физический том состоит из блоков определённого размера, называемых экстентами. Количество и размер физических экстентов определяют размер физического тома;
- **LE (Logical Extent)** – каждый логический том состоит из блоков определённого размера, кратного размеру физического экстента.

Процедура создания логических томов состоит из создания физических томов и их объединения в группы. Чтобы получить информацию о созданных физических томах, воспользуйтесь командой **`pvscan`**, команды **`vgscan`** и **`lvscan`** выводят информацию о группах томов и логических томах, соответственно.

Чтобы создать на диске/разделе диска структуру физического тома, воспользуйтесь командой pvcreate, в случае, когда вы работаете с разделом, сначала установите тип раздела в значение 0x8e (командой **`fdisk`**):

```
# pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created
# pvcreate /dev/sdb2
Physical volume "/dev/sdb2" successfully created
```

Теперь, когда есть два физических тома, их необходимо объединить в группу с помощью команды **vgcreate**:

```
# vgcreate vg00 /dev/sdb1 /dev/sdb2
Volume group "vg00" successfully created
```

Теперь перейдём к созданию логических томов, объектов, с которыми и будет проводиться дальнейшая работа операционной системы. Воспользуемся командой **lvcreate**:

```
# lvcreate -L1000 -nlv1 vg00
Logical volume "lv1" created
```

Следующим шагом является активация группы томов командой **vgchange**:

```
# vgchange -ay
```

На этом этапе мы уже имеем один логический том и можем получить о нём информацию командой **lvdisplay**. Но этот том не приспособлен для работы, так как не имеет никакой файловой системы, давайте создадим её:

```
# mkfs -t ext3 /dev/vg00/lv1
```

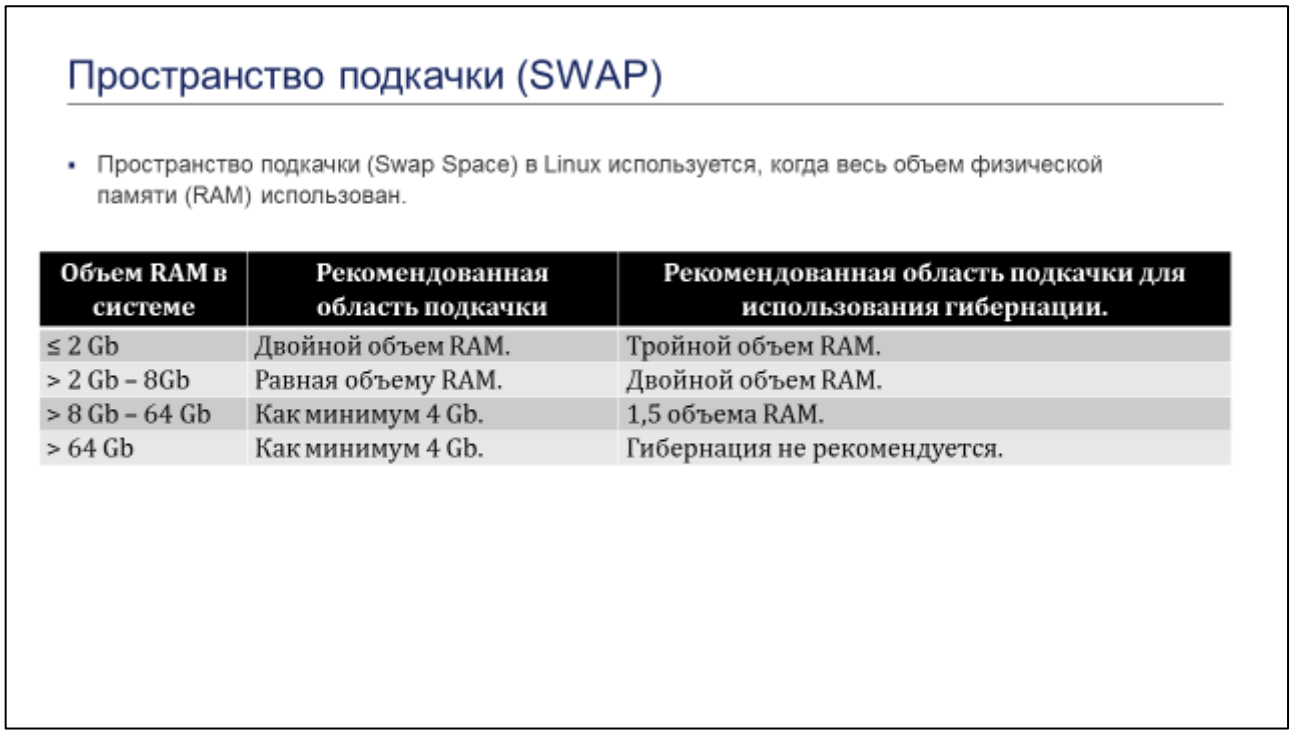
Случаются ситуации, когда свободное место логического тома заканчивается и его необходимо расширить, для этой операции существует команда lvextend:

```
# lvextend -L+500M /dev/vg00/lv1
Logical volume lv1 successfully resized
```

Но этой командой мы увеличили только размер тома, но не размер файловой системы, для изменения размера файловой системы воспользуемся командой **resize2fs**, перед эти выполнив проверку файловой системы:

```
# e2fsck -f /dev/vg00/lv1
# resize2fs /dev/vg00/lv1
```

Для уменьшения размера тома предназначена команда **lvreduce**.



Пространство подкачки (Swap Space) в Linux используется, когда весь объем физической памяти (RAM) использован. Если системе требуется дополнительная память, а физическая память (RAM) уже исчерпана, неактивные страницы перемещаются в область подкачки. Несмотря на то что подкачка может помочь машинам с небольшим объемом RAM, она не является заменой физической памяти. Пространство подкачки располагается на жестких дисках, которые обладают более медленной скоростью доступа чем физическая память. Пространство подкачки может быть выделено в раздел подкачки (рекомендуется), файл подкачки мом комбинацию из разделов и файлов подкачки.

ПРИМЕЧАНИЕ

Btrfs не поддерживает пространство подкачки.

Последние годы рекомендованный объем пространства подкачки увеличивался линейно, вместе с объемом RAM в системе. Однако, современные системы часто содержат сотни гигабайтов RAM. В результате рекомендованное пространство подкачки рассматривается как функция нагрузки на системную память, а не как системная память.

В таблице ниже предоставлены рекомендованные размеры разделов подкачки в зависимости от объема RAM в системе и возможности использования гибернации на системе. Рекомендованный размер раздела подкачки вычисляется автоматически в процессе установки. Однако, чтобы разрешить гибернацию на системе, необходимо вручную отредактировать размер раздела подкачки на этапе разметки диска.

Рекомендации в таблице ниже особенно важны для систем с с объемом памяти 1 Gb и меньше. Некорректное выделение пространства подкачки на системе, может стать причиной нестабильной работы или невозможности запуска установленной системы.

Объем RAM в системе	Рекомендованная область подкачки	Рекомендованная область подкачки для использования гибернации.
≤ 2 Gb	Двойной объем RAM.	Тройной объем RAM.
> 2 Gb – 8Gb	Равная объему RAM.	Двойной объем RAM.
> 8 Gb – 64 Gb	Как минимум 4 Gb.	1,5 объема RAM.
> 64 Gb	Как минимум 4 Gb.	Гибернация не рекомендуется.

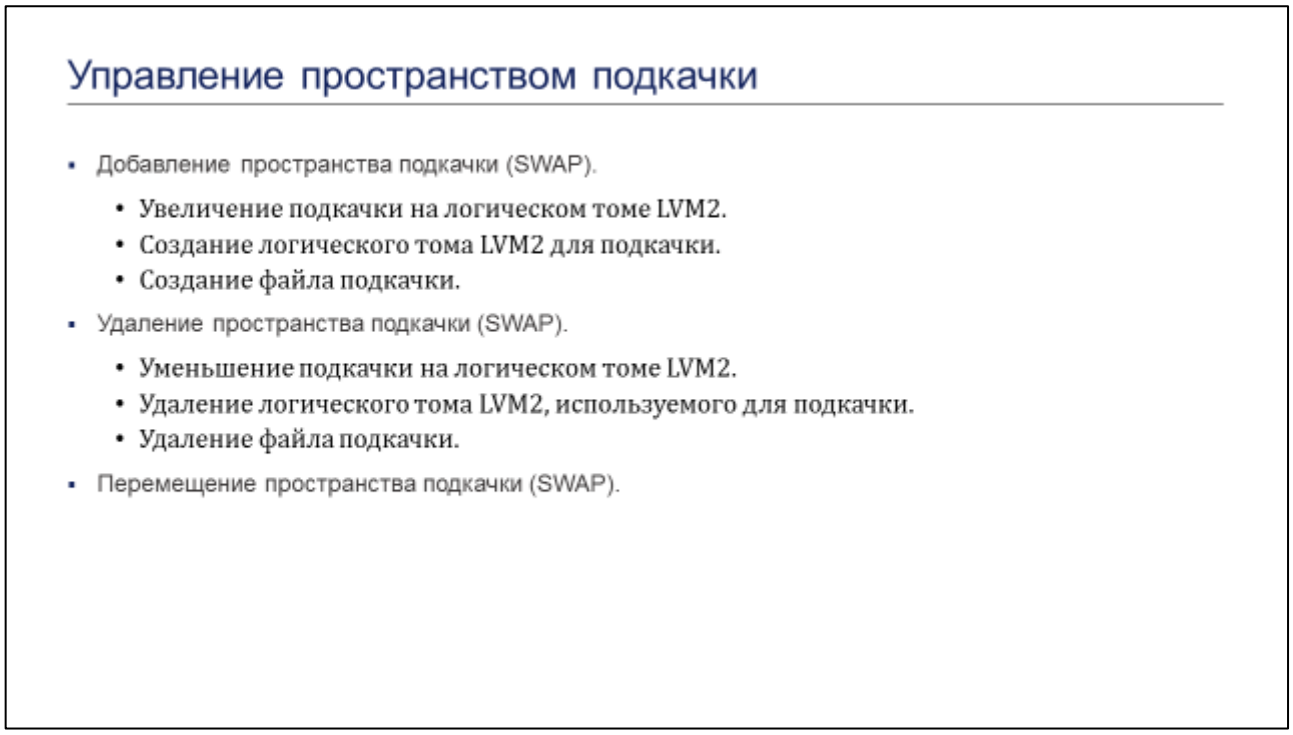
На границе между каждым диапазоном, указанным в таблице выше, основной выбор заключается в использовании или не использовании режима гибернации на системе. Если системные ресурсы позволяют, то увеличение объема пространства подкачки, может увеличить производительность системы. Область подкачки, как минимум в 100 Gb рекомендуется для систем со 140 и более логическими процессорами или 3 и более Tb физической памяти.

Распределение области подкачки по нескольким устройствам хранения также увеличивают производительность пространства подкачки, особенно на системах с быстрыми дисками, котроллерами и интерфейсами.

ПРИМЕЧАНИЕ

Файловые системы и тома LVM2 назначенные пространству подкачки не должны использоваться в процессе модификации. Любая попытка изменить подкачку приведет к ошибке процесса или ядра, использующую пространство подкачки. Используйте команду **`free`** и **`cat /proc/swaps`** для проверки того, сколько и где подкачки используется.

Вы должны изменять область подкачки, когда система загружена в аварийном режиме (Rescue Mode).



В некоторых случаях необходимо увеличить объем пространства подкачки после установки системы. Например, в случае увеличения объема RAM на вашей системе с 1 Gb до 2 Gb, при этом объем подкачки сохраняется в размере 2 Gb. Вы можете получить преимущества от увеличения области подкачки до 4 Gb на системах с активным использованием памяти или приложениями, требующими большего объема памяти.

У вас есть три опции: создать новый раздел подкачки, создать файл подкачки или увеличить область подкачки на существующем логическом томе LVM2. В случае наличия последнего рекомендуется увеличивать логический том.

Увеличение подкачки на логическом томе LVM2.

По умолчанию RedOS 7 использует все доступное пространство в процессе установки. В таком случае вам необходимо добавить дополнительный физический том (Physical Volume) в группу томов (Volume Group) на которой размещается область подкачки. После добавления дополнительного хранилища в группу томов, можно расширять область подкачки.

Процедура расширения подкачки на логическом томе LVM2:

1. Отключение подкачки на логическом томе:

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. Увеличение размера логического тома на 2 Gb:

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. Форматирование нового пространства подкачки:

```
# mkswap /dev/VolGroup00/LogVol01
```

4. Включение подкачки на увеличенном логическом томе:

```
# swapon -v /dev/VolGroup00/LogVol01
```

Чтобы проверить, что логический том и область подкачки увеличились используйте команды **free** или **cat /proc/swaps**.

Создание логического тома LVM2 для подкачки.

Для добавления логического тома подкачки в группе томов, используйте следующую процедуру:

1. Создание логического тома, размером 2

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. Форматирование нового пространства подкачки:

```
# mkswap /dev/VolGroup00/LogVol02
```

3. Добавление новой записи в /etc/fstab:

```
/dev/VolGroup00/LogVol02   swap   swap   defaults   0 0
```

4. Включение подкачки на новом логическом томе LVM2:

```
# swapon -v /dev/VolGroup00/LogVol02
```

Чтобы проверить, что логический том и область подкачки увеличились используйте команды **free** или **cat /proc/swaps**.

Создание файла подкачки.

Для добавления файла подкачки, используйте следующую процедуру:

1. Определите необходимый размер файла подкачки в мегабайтах и умножьте на 1024 для определения необходимого числа блоков. Например, для файла подкачки, размером 64 Mb, потребуется 65536 блоков по 1 Kb каждый.
2. При помощи команды dd создайте необходимый файл:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

3. Отформатируйте файл подкачки:

```
# mkswap /swapfile
```

4. Настройте безопасность файла подкачки:

```
# chmod 0600 /swapfile
```

5. Включите использование файла подкачки:

```
# swapon /swapfile
```

6. Настройте автоматический запуск файла подкачки в процессе загрузки системы:

```
/swapfile   swap   swap   defaults   0 0
```

Чтобы проверить, что файл подкачки используется используйте команды **free** или **cat /proc/swaps**.

Удаление пространства подкачки (SWAP).

Иногда бывает необходимо сократить область подкачки после установки. Например, если вы уменьшили объем RAM на вашей системе с 1 Gb до 512 Mb, в таком случае 2 GB области подкачки остается назначено системе. Вы можете получить преимущества от сокращения области подкачки до 1 Gb на системе, вместо простаивающего пространства подкачки размером в 2 Gb.

У вас есть три опции: удалить весь логический том LVM, используемый для подкачки, удалить файл подкачки или уменьшить область подкачки на существующем логическом томе LVM2.

Уменьшение подкачки на логическом томе LVM2.

Для уменьшения логического тома LVM2, используемого для подкачки, воспользуйтесь следующей процедурой:

1. Отключите подкачку на указанном логическом томе LVM2:

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. Уменьшите логический том на 1 Gb:

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. Отформатируйте новое пространство подкачки:

```
# mkswap /dev/VolGroup00/LogVol01
```

4. Включите подкачку на уменьшенном логическом томе LVM2:

```
# swapon -v /dev/VolGroup00/LogVol01
```

Чтобы проверить, что логический том и область подкачки уменьшились используйте команды **free** или **cat /proc/swaps**.

Удаление логического тома LVM2, используемого для подкачки.

Для удаления логического тома LVM2, используемого для подкачки, воспользуйтесь следующей процедурой:

1. Отключите подкачку на связанном логическом томе:

```
# swapoff -v /dev/VolGroup00/LOgVol02
```

2. Удалите логический том LVM2:

```
# lvremove /dev/VolGroup00/LogVol02
```

3. Удалите или закоментируйте запись из файла /etc/fstab:

```
/dev/VolGroup00/LogVol02   swap   swap   defaults   0 0
```

Чтобы проверить, что логический том удален и область подкачки уменьшилась используйте команды **free** или **cat /proc/swaps**.

Удаление файла подкачки.

Для удаления файла подкачки, воспользуйтесь следующей процедурой:

1. Отключите использование файла для подкачки:

```
# swapoff -v /swapfile
```

2. Удалите строку из файла /etc/fstab:

```
/swapfile   swap   swap   defaults   0 0
```

3. Удалите файл подкачки:

```
# rm /swapfile
```

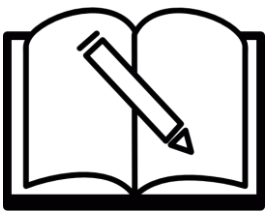
Перемещение пространства подкачки (SWAP).

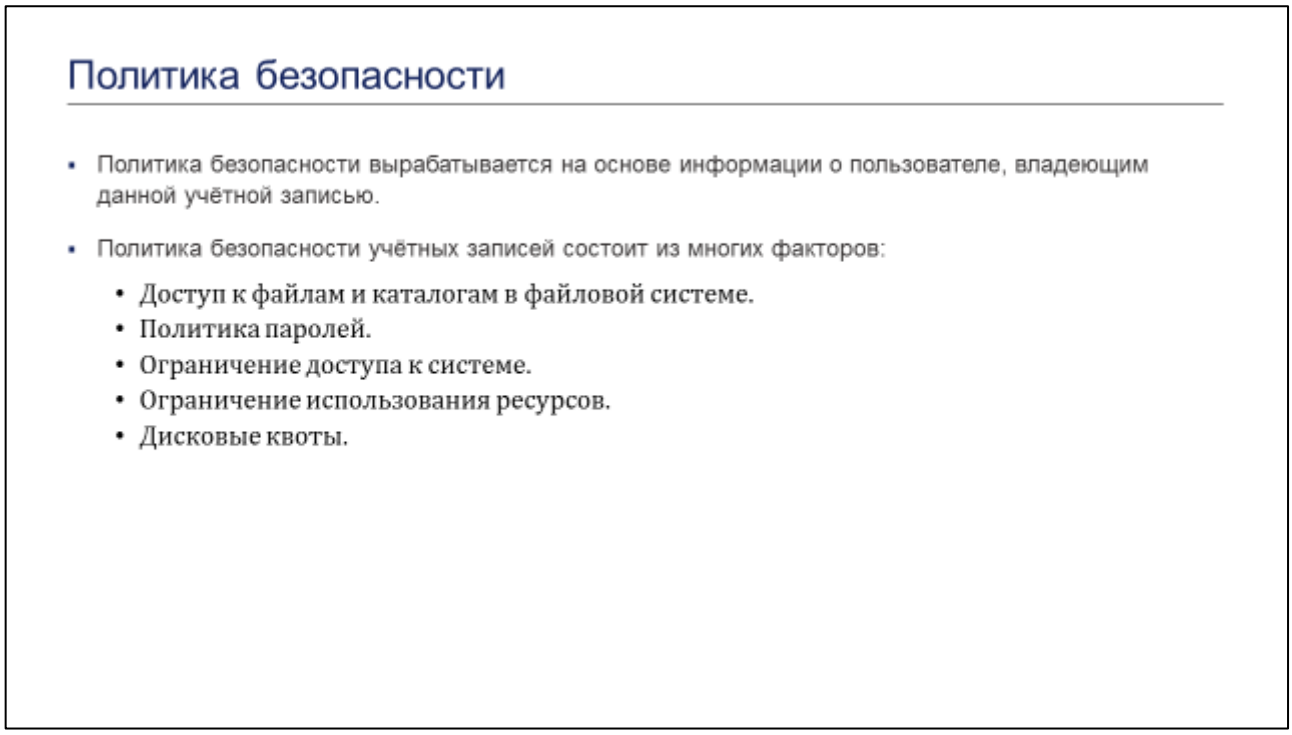
Для перемещения пространства подкачки используйте шаги по добавлению нового пространства подкачки и удалению старого.

Модуль 4. Управление пользователями в РЕД ОС

В этом модуле рассматриваются:

- Локальные пользователи и группы
- Получение привилегий
- Управление политиками паролей
- Квоты на файловой системе

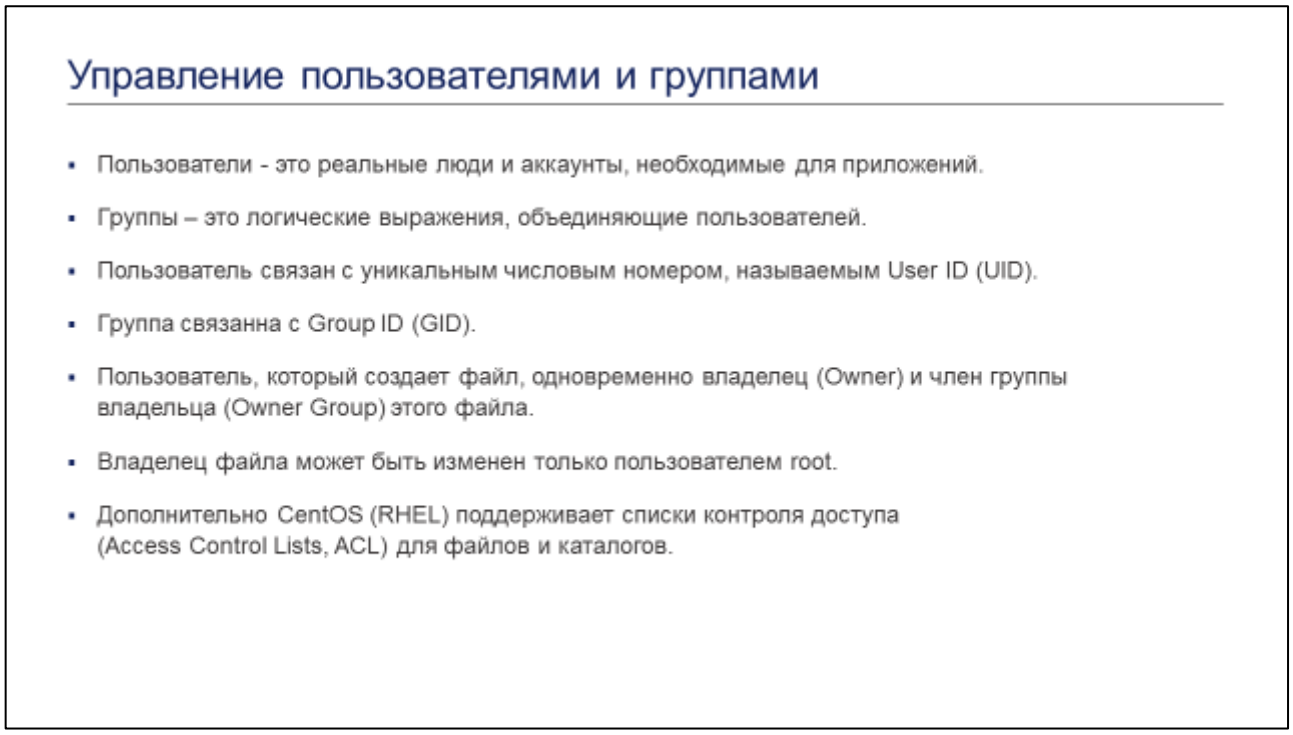




При управлении учётными записями, необходимо помнить о том, что у всех пользователей полномочия должны иметь четкие границы. Тщательное планирование политики учётных записей – один из китов, составляющих безопасность вычислительной системы.

Основные аспекты, на которые следует обратить внимание при планировании политики безопасности:

- **Ограничение доступа к файлам и каталогам файловой системы.** Как вы уже заметили, в операционной системе Linux пользователи имеют доступ не ко всем файлам. Более того, каждый файл может быть доступен учётной записи, за которой не закреплён пользователь (системной записи). Файл – основа операционной системы. Если простой пользователь получит доступ к файлу, в котором хранятся пароли, то такую систему нельзя назвать безопасной.
- **Парольная политика** очень важна для стабильной и безопасной системы. Использование коротких паролей, являющихся простыми словами небезопасно. Хороший пароль состоит из набора буков (в разном регистре) и цифр, чем длиннее такой пароль, тем он "прочнее", что затруднит процесс его взлома.
- **Ограничение доступа к системе.** Любой пользователь должен иметь доступ к той системе, которая ему необходима, никогда не давайте пользователю излишних привилегий. Иногда необходимо ограничить доступ по времени входа, согласитесь, странно, если секретарь захочет войти в систему в 3 часа ночи.
- **Ограничения, касающиеся системных ресурсов**, позволят не допустить большого расходования ресурсов пользователем.
- **Дисковые квоты** позволят ограничить максимальный объём дискового пространства, доступного пользователю.



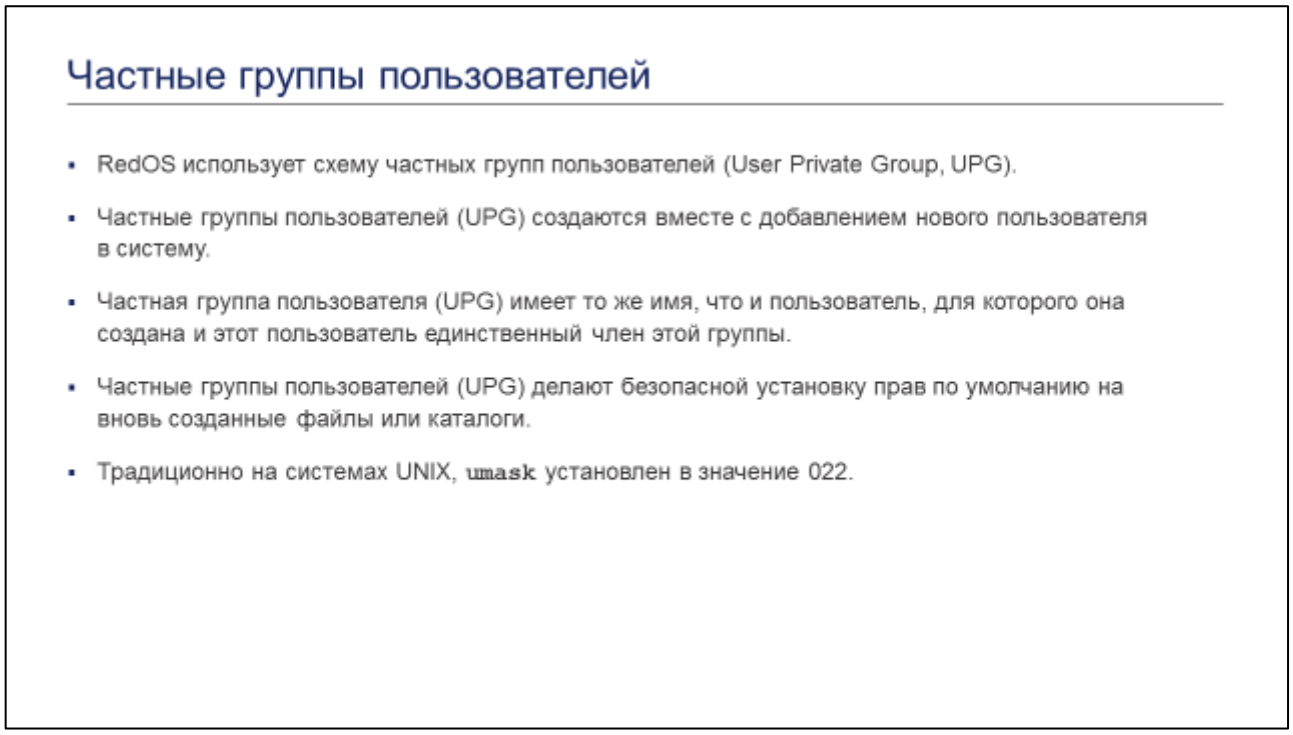
Управление пользователями и группами – это ключевой элемент системного администрирования RedOS.

Описание пользователей и групп.

Пользователи одновременно могут быть и реальным людьми (аккаунтами, привязанными к физическим людям) и аккаунтами, которые необходимы для использования определенными приложениями. Группы – это логические выражения организации, объединяющие пользователей вместе для общих задач. Пользователи, состоящие в группе, могут одновременно читать, писать или выполнять файлы, принадлежащие группе.

Каждый пользователь связан с уникальным числовым номером, называемым User ID (UID). По аналогии каждая группа связанна с Group ID (GID). Пользователь, который создает файл, одновременно владелец (Owner) и член группы владельца (Owner Group) этого файла. Файлу назначаются разные комбинации прав чтение (Read), запись (Write) и выполнение (Execute) для владельца, группы владельца и всех остальных. Владелец файла может быть изменен только пользователем root, права доступа могут быть изменены только пользователем root и владельцем файла.

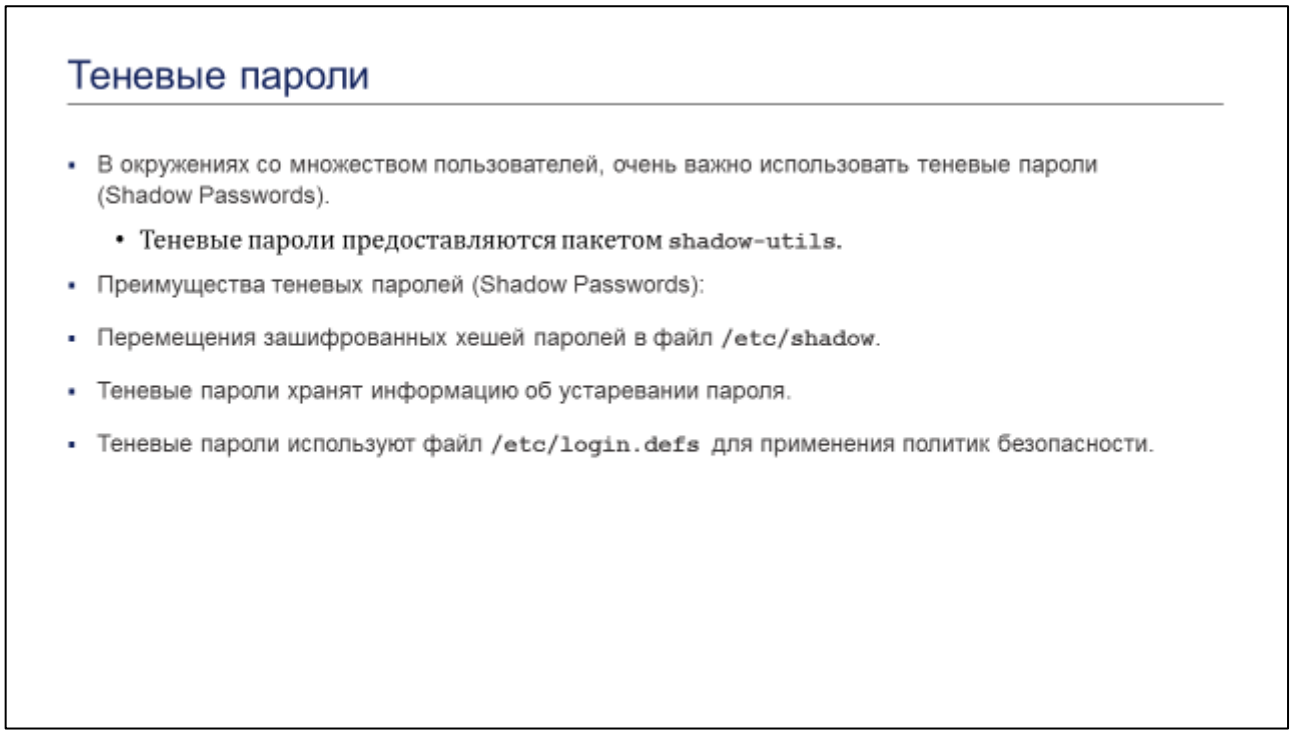
Дополнительно RedOS поддерживает списки контроля доступа (Access Control Lists, ACL) для файлов и каталогов, которые позволяют назначать права за пределами стандартных пользователя, группы и всех остальных.



RedOS использует схему частных групп пользователей (User Private Group, UPG), которая упрощает управление группами UNIX. Частные группы пользователей (UPG) создаются вместе с добавлением нового пользователя в систему. Частная группа пользователя (UPG) имеет то же имя, что и пользователь, для которого она создана и этот пользователь единственный член этой группы.

Частные группы пользователей (UPG) делают безопасной установку прав по умолчанию на вновь созданные файлы или каталоги, позволяя пользователю и его частной группе вносить изменения в файл или каталог.

Параметры, которые определяют права к вновь созданным файлам или каталогам называются **umask** и настраиваются в файле **/etc/bashrc**. Традиционно на системах UNIX, **umask** установлен в значение **022**, что позволяет только пользователям, которые создали файл или каталог вносить изменения. В рамках этой схемы, все остальные пользователи, включая членов группы создателя, не имеют права вносить какие-либо изменения. Однако, в рамках схемы UPG, данная защита группы (Group Protection) не обязательна, так как каждый пользователь имеет собственную частную группу.



В окружениях со множеством пользователей, очень важно использовать теневые пароли (Shadow Passwords), предоставляемые пакетом **shadow-utils** для расширения безопасности файлов системной аутентификации. Для этих целей, программа установщик включает теневые пароли по умолчанию.

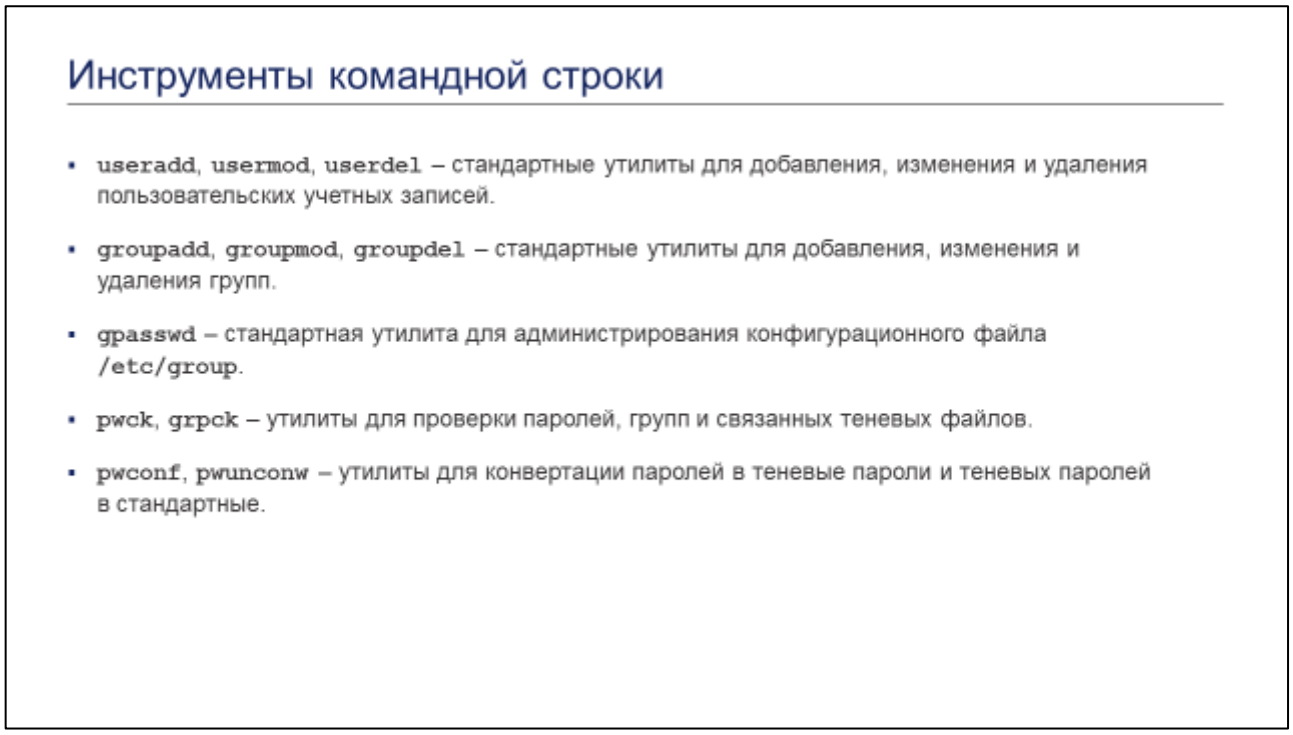
Список преимуществ теневых паролей над традиционным хранением паролей в системах UNIX:

- Теневые пароли улучшают безопасность системы, за счет перемещения зашифрованных хешей паролей в файл **/etc/shadow** (доступен для чтения только для пользователя root) из незашифрованного вида в файле **/etc/passwd**.
- Теневые пароли хранят информацию об устаревании пароля.
- Теневые пароли позволяют файлу **/etc/login.defs** для применения политик безопасности.

Большинство утилит, предоставляемых пакетом **shadow-utils**, работают корректно вне зависимости от того включено или отключено использование теневых паролей. Однако, с тех пор как информация об устаревании пароля эксклюзивно хранится в файле **/etc/shadow**, любые команды, которые создают или изменяют информацию об устаревании пароля не работают.

Список утилит, которые не работают, если не включены теневые пароли:

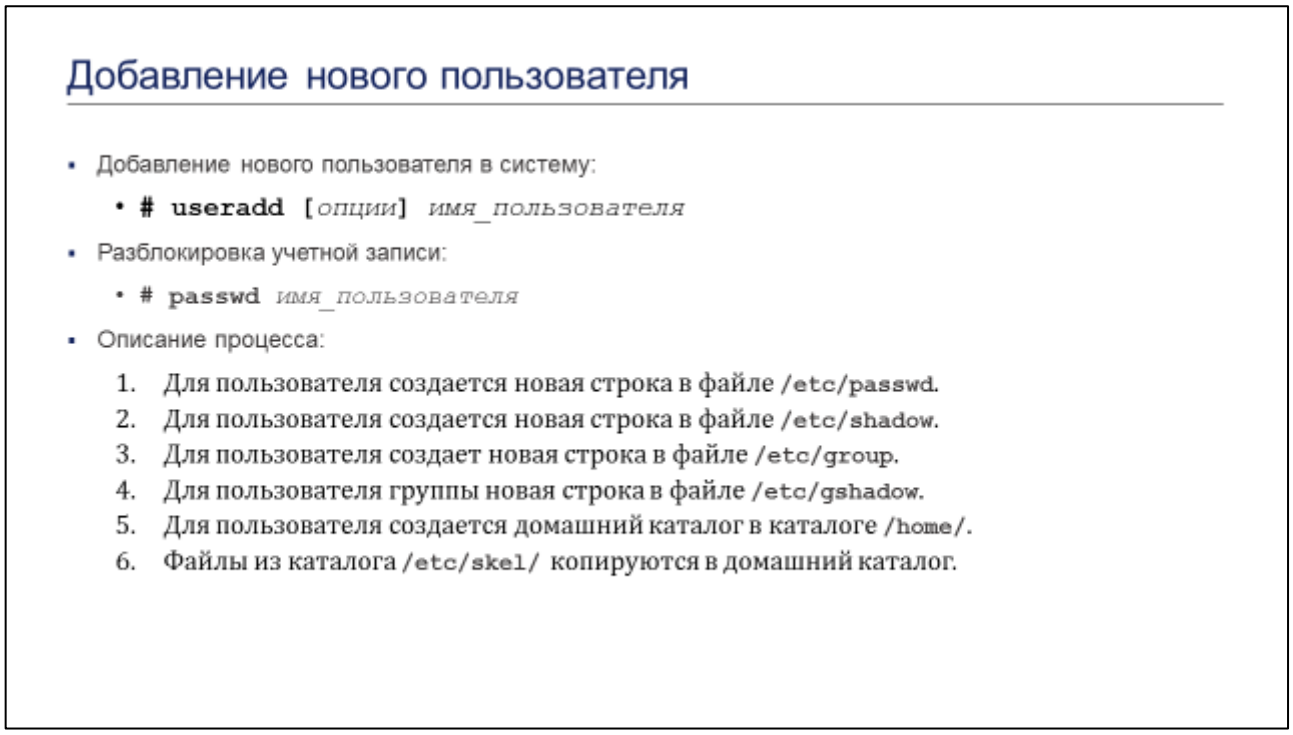
- Утилита **chage**.
- Утилита **gpasswd**.
- Команда **usermod** с опциями **-e** или **-f**.
- Команда **useradd** с опциями **-e** или **-f**.



Наиболее простой путь управления пользователями и группами в RedOS – это использование графических инструментов. Однако, если вы предпочитаете инструменты командной строки или на системе не установлена X Window System, вы можете использовать утилиты командной строки.

Утилиты командной строки для управления пользователями и группами.

Утилиты	Описание
useradd, usermod, userdel	Стандартные утилиты для добавления, изменения и удаления пользовательских учетных записей.
groupadd, groupmod, groupdel	Стандартные утилиты для добавления, изменения и удаления групп.
gpasswd	Стандартная утилита для администрирования конфигурационного файла **/etc/group**.
pwck, grpck	Утилиты для проверки паролей, групп и связанных теневых файлов.
pwconf, pwunconw	Утилиты для конвертации паролей в теневые пароли и теневых паролей в стандартные.



Для добавления нового пользователя в систему, ведите следующую команду, от имени пользователя root:

```
# useradd [ОПЦИИ] ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

По умолчанию команда **useradd** создает заблокированную учетную запись пользователя. Для разблокировки учетной записи, выполните следующую команду от имени пользователя root:

```
# passwd ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

Опционально можно установить политику устаревания пароля.

Таблица популярных опций командной строки команды **useradd**.

Опция	Описание
-c *'комментарий'*	Комментарий, как правило используют для указания полного имени.
-d *домашний_каталог*	Домашний каталог, который будет использоваться вместо **/home/***ИМЯ_ПОЛЬЗОВАТЕЛЯ*.
-e *дата*	Дата для учетной записи, в которую она будет отключена в формате *ГГГГ-ММ-ДД*.
-f *дни*	Число дней, после истечения срока действия пароля, когда учетная запись будет отключена. Если указано значение **0**, учетная запись отключается сразу после истечения срока действия пароля. Если указано значение **-1**, учетная запись не будет отключаться после истечения срока действия пароля.
-g *имя_группы*	Имя группы или GID, указывает группу пользователя по умолчанию. Группа должна быть создана предварительно.
-G *список_групп*	Список дополнительных групп или числовых значений GID (помимо группы по умолчанию), в которые необходимо добавить пользователя. Группы должны быть созданы предварительно.
-m	Создает домашний каталог, если он не существует.
-M	Не создает домашний каталог.
-N	Не создает частной группы для пользователя.

-p *пароль*	Пароль, зашифрованный при помощи **crypt**.
-r	Создает системную учетную запись с UID меньше 500 и без домашнего каталога.
-s	Оболочка пользователя, по умолчанию: **/bin/bash**.
-u *UID*	Указывает UID пользователя, должен быть уникальным и больше чем **499**.

Описание процесса.

Следующие шаги иллюстрируют действия, которые происходят на системе с включенными теневыми паролями, когда вызывается команда **useradd student**.

1. Для пользователя **student** создается новая строка в файле **/etc/passwd**:

```
student:x:501:501::/home/juan:/bin/bash
```

В строке содержится 7 полей:

1	**student**	Имя пользователя (username).
2	**x**	Обозначает, что система использует теневые пароли.
3	**501**	UID, в RedOS значения UID до **500** зарезервированы для системного использования и не должны назначаться пользователям.
4	**501**	GID, в RedOS значения GID до **500** зарезервированы для системного использования и не должны назначаться пользователям.
5	*пусто*	опциональная информация (GECOS), как правило используется для указания полного имени или какой-либо дополнительной информации.
6	**/home/student**	домашний каталог.
7	**/bin/bash**	оболочка по умолчанию.

2. Для пользователя **student** создается новая строка в файле **/etc/shadow**:

```
student:!!:14798:0:99999:7:::
```

В строке содержится 9 полей:

1	**student**	Имя пользователя
2	**!!**	Зашифрованный пароль, два восклицательных знака (**!!**) – блокируют учетную запись.
3	**14798**	Дата последнего изменения пароля.
4	**0**	Минимальный возраст пароля.
5	**99999**	Максимальный возраст пароля.
6	**7**	Период предупреждения об истечении пароля.
7	*ПУСТО*	Период неактивного пароля.
8	*ПУСТО*	Дата устаревания учетной записи.
9	*ПУСТО*	Зарезервированное поле.

3. Для пользователя **student** создает новая строка в файле **/etc/group**, содержащая информацию о частной группе пользователя:

```
student:x:501:
```

В строке содержится 4 поля:

1	**student**	Имя группы.
2	**x**	Обозначает, что система использует теневые пароли
3	**501**	GID, в RedOS значения GID до 500 зарезервированы для системного использования и не должны назначаться пользователям.
4	*ПУСТО*	Список членов группы (указываются username), за исключением владельца частной группы.

4. Для пользователя группы новая строка в файле **/etc/gshadow**:

```
student:!::
```

В строке содержится 4 поля:

1	**student**	Имя группы.
2	**!**	Зашифрованный пароль, восклицательный знак обозначает что группа заблокирована.
3	*ПУСТО*	Администраторы группы.
4	*ПУСТО*	Члены группы.

5. Каталог для пользователя **student** создается в каталоге **/home/**.

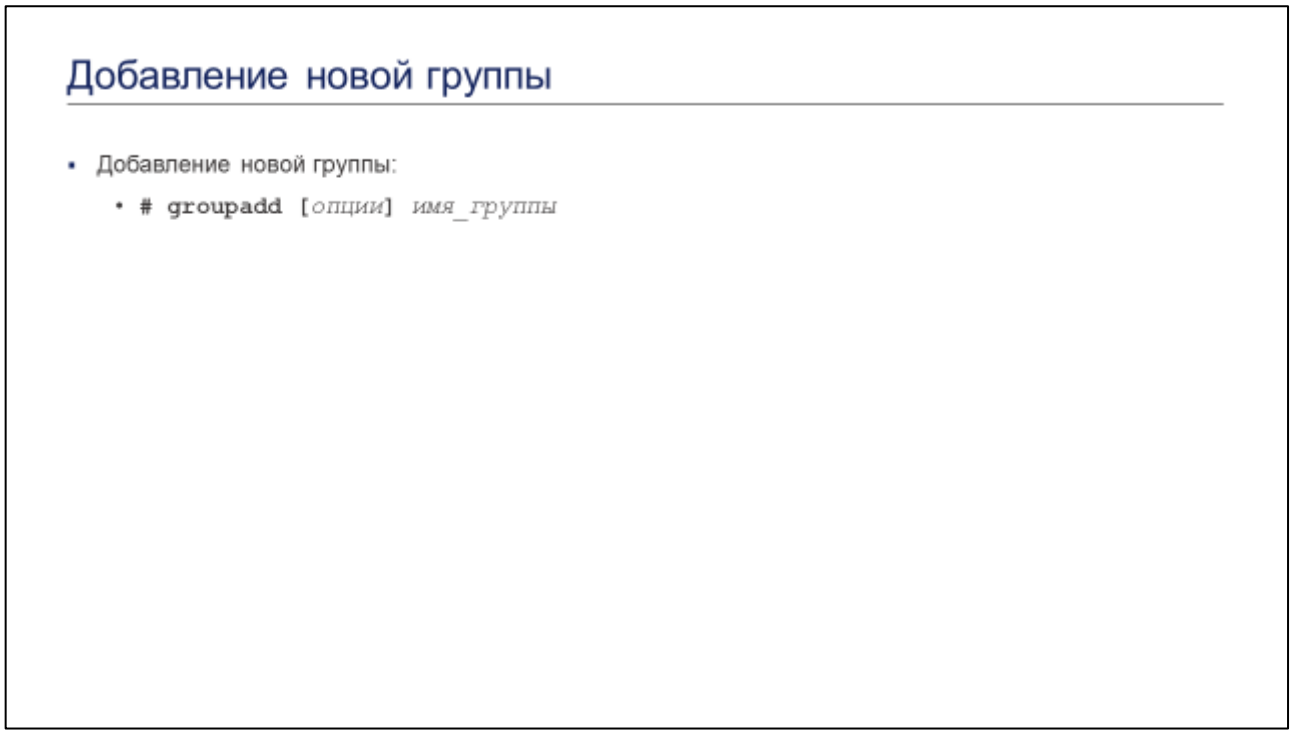
```
drwx------. 4 juan juan 4096 Mar 3 18:23 student
```

Этот каталог принадлежит пользователю **student** и группе **student**. Права на чтение, запись и выполнение предоставлены только пользователю student. Все остальные права запрещены.

6. Файлы из каталога **/etc/skel/** (который содержит настройки пользователя по умолчанию) копируются в созданный на предыдущем шаге домашний каталог пользователя.

```
drwx------. 4 student student 4096 Mar 3 18:23 .
drwxr-xr-x. 5 root root 4096 Mar 3 18:23 ..
-rw-r--r--. 1 student student 18 Jun 22 2010 .bash_logout
-rw-r--r--. 1 student student 176 Jun 22 2010 .bash_profile
-rw-r--r--. 1 student student 124 Jun 22 2010 .bashrc
drwxr-xr-x. 2 student student 4096 Jul 14 2010 .gnom e2
drwxr-xr-x. 4 student student 4096 Nov 23 15:09 .mozilla
```

На этом этапе заблокированная учетная запись **student** существует на системе. Для ее активации необходимо назначить пароль, при помощи команды **passwd**, опционально можно задать дополнительные параметры пароля.

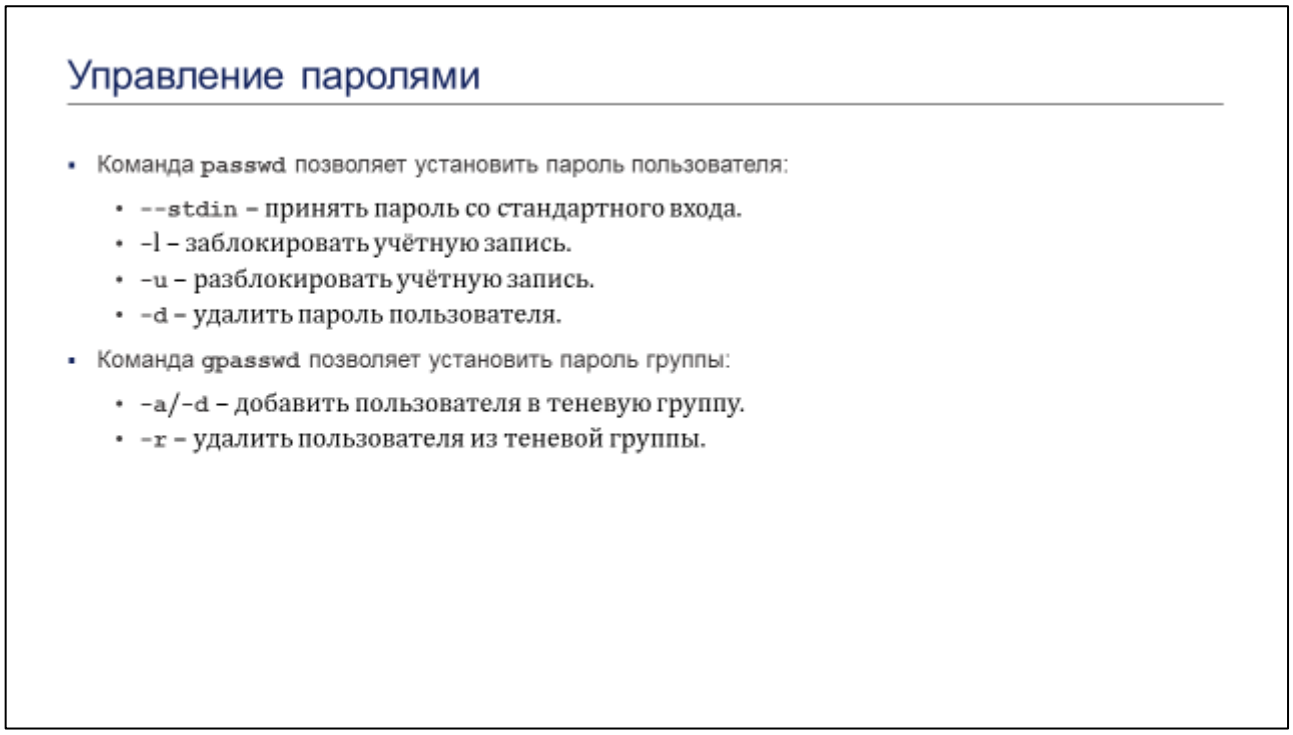


Для добавления новой группы на систему, введите следующую команду от имени пользователя root:

```
# groupadd [ОПЦИИ] ИМЯ_ГРУППЫ
```

Таблица популярных опций командной строки команды **groupadd**.

Опция	Описание
-f, **--force**	Комментарий, как правило используют для указания полного имени.
-g *GID*	GID, в CentOS (RHEL) значения GID до 500 зарезервированы для системного использования и не должны назначаться пользователям.
-K, **--key** *ключ*=*значение*	Переписывает значения по умолчанию, определенные в файле `/etc/login.defs`.
-o, **--non-unique**	Позволяет создавать дубликаты групп с неуникальным GID.
-p, --password	Указывает зашифрованный пароль для новой группы.
-r	Создает системную группу с GID меньше 500.



Пароли – одно из самых уязвимых мест в системе безопасности IT-инфраструктуры. Администратор может организовать очень защищённую структуру сети, но пользователь, имеющий высокие привилегии и использующий "слабый" пароль, может сохранить огромную уязвимость всей системы. Именно по этой причине, хороший администратор обязан сделать всё для того, чтобы пользователи не могли использовать "слабые" пароли.

Какие действия необходимо принять, чтобы пароли пользователей не ослабляли систему безопасности:

- Планирование и создание парольной политики, запрещающей установление "слабых" паролей;
- Создание паролей при помощи генераторов паролей;
- Периодический аудит паролей всех учётных записей.

Пароли учётных записей пользователей задаются с помощью программы **passwd**:

```
# passwd student
```

Программа **passwd** используется не только для задания пароля, но и для блокирования/разблокирования учётных записей пользователей:

```
# passwd -l student
# passwd -u student
```

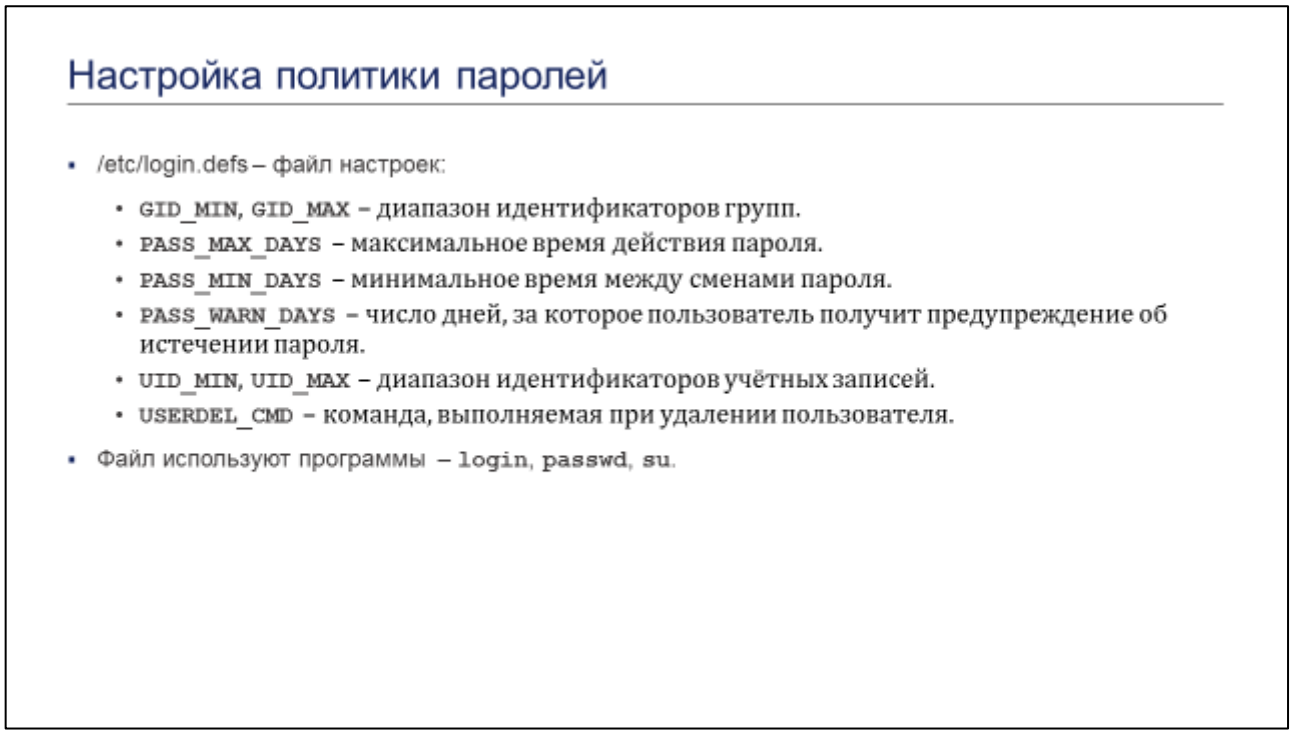
Команду passwd можно использовать в сценариях командного процессора, используя параметр **--stdin**, который указывает, что пароль необходимо принять на стандартный канал входа:

```
# echo password > passwd --stdin student
```

Как говорилось ранее, пароли могут быть заданы не только для учётных записей пользователей, но и для учётных записей групп. Задать пароль учётной записи группы можно с помощью команды **gpasswd**:

```
# gpasswd marketing
```

Также, **gpasswd** используется для добавления учётных записей пользователей в теневую группу.



Файл **/etc/login.defs** используется для настройки служб, использующих систему теневых паролей. Это текстовый файл, каждая строка описывает один параметр.

С помощью этого файла задаётся парольная политика, политика учётных записей и многое другое:

GID_MIN/GID_MAX – определение диапазона идентификаторов, из которого будут браться номера для создания новых учётных записей.

UID_MIN/UID_MAX – определение диапазона идентификаторов, из которого будут браться номера для создания новых учётных записей.

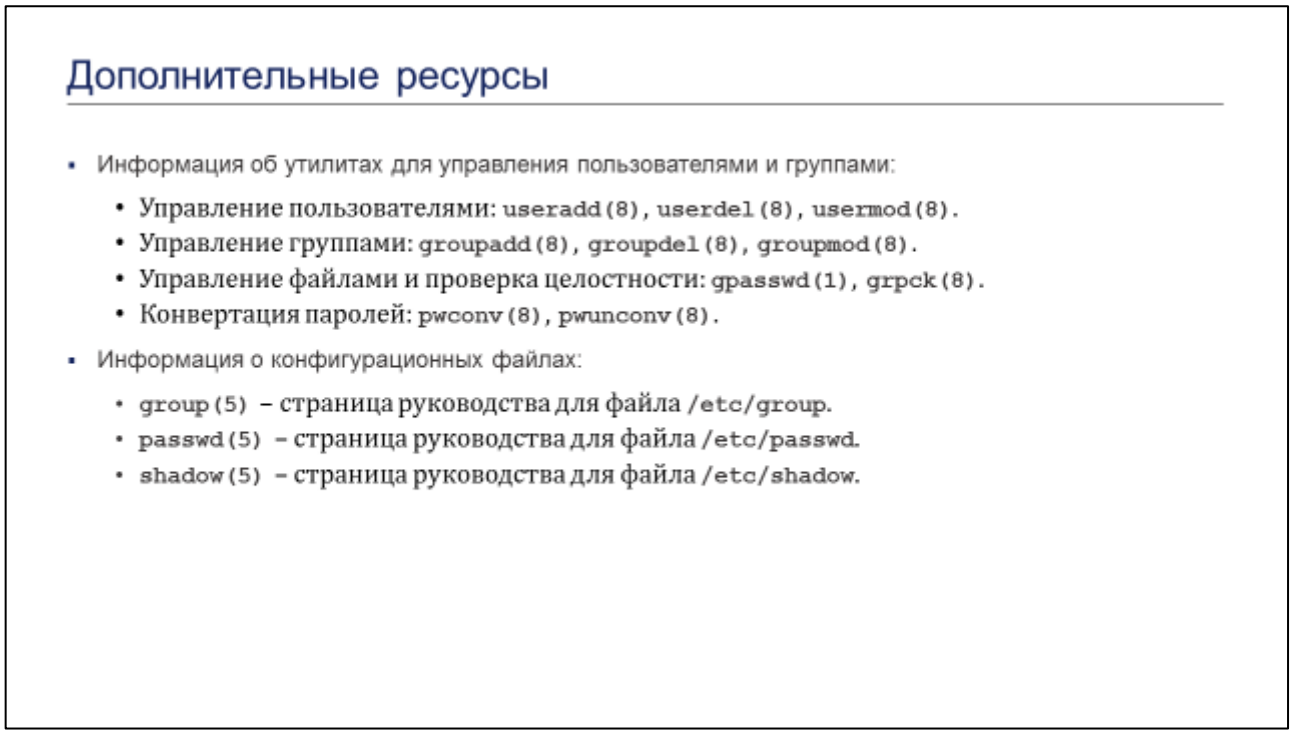
PASS_MAX_DAYS – определяет максимальное время актуальности пароля.

PASS_WARN_DAYS – определяет число дней, в течение которых пользователю будет выдаваться предупреждение о скором блокировании учётной записи.

PASS_MIN_LEN – минимальная длина пароля.

USERDEL_CMD – указание команды, которую необходимо выполнить при удалении пользователя. Это очень полезный параметр, позволяющий удалять все "хвосты" автоматически.

Дополнительная безопасность обеспечивается библиотекой **cracklib**, которая позволяет проверить устанавливаемый пароль с помощью словарей.



Для получения дополнительной информации об управлении пользователями и группами, воспользуйтесь следующими ресурсами:

Установленная документация.

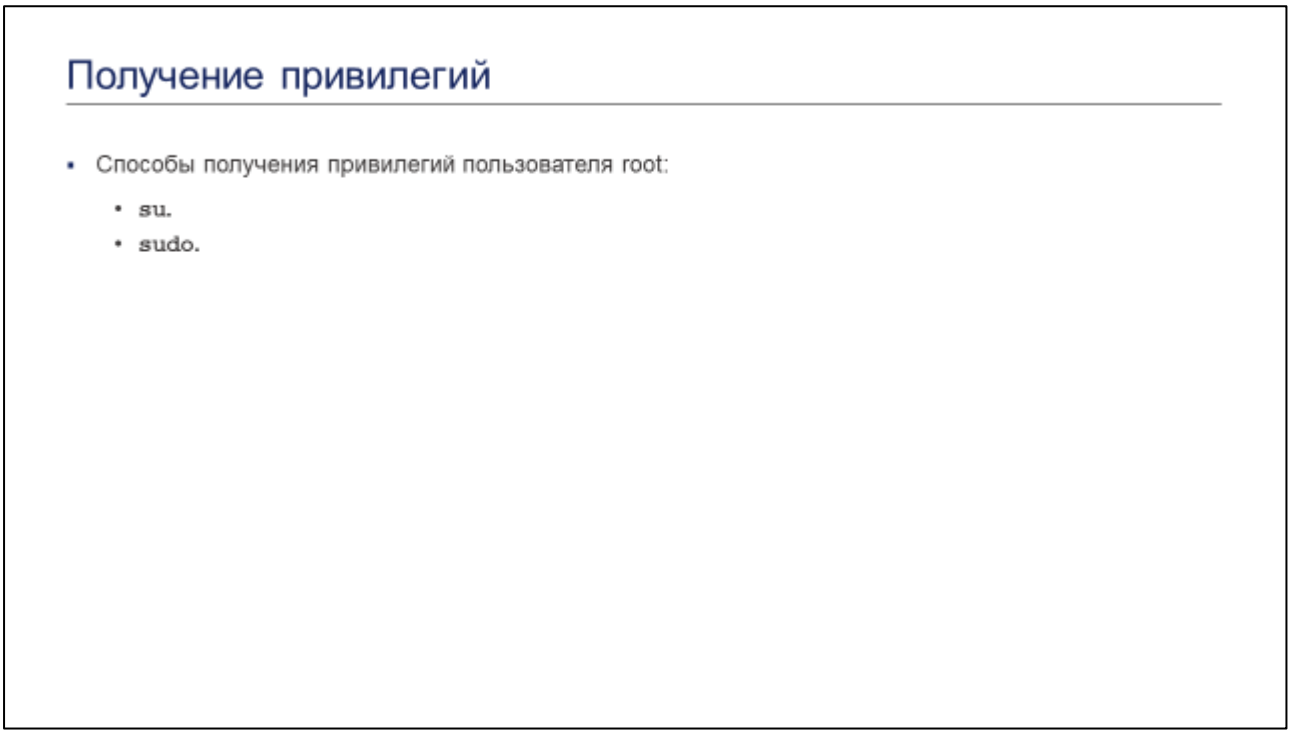
Для получения информации об утилитах для управления пользователями и группами в RedOS, воспользуйтесь следующими страницами руководства:

- **useradd(8)** – страница руководства для команды **useradd**, содержит информацию о создании новых пользователей.
- **userdel(8)** – страница руководства для команды **userdel**, содержит информацию об удалении пользователей.
- **usermod(8)** – страница руководства для команды **usermod**, содержит информацию об изменении пользователей.
- **groupadd(8)** – страница руководства для команды **groupadd**, содержит информацию о создании новых групп.
- **groupdel(8)** – страница руководства для команды **groupdel**, содержит информацию об удалении групп.
- **groupmod(8)** – страница руководства для команды **groupmod**, содержит информацию об изменении групп.
- **gpasswd(1)** – страница руководства для команды **gpasswd**, содержит информацию об управлении файлом **/etc/group**.
- **grpck(8)** – страница руководства для команды **grpck**, содержит информацию о проверке целостности файла **/etc/group**.
- **pwck(8)** – страница руководства для команды gwck, содержит информацию о проверке целостности файлов **/etc/passwd** и **/etc/shadow**.
- **pwconv(8)** – страница руководства для команды **pwconw**, содержит информацию о конвертации стандартных паролей в теневые.

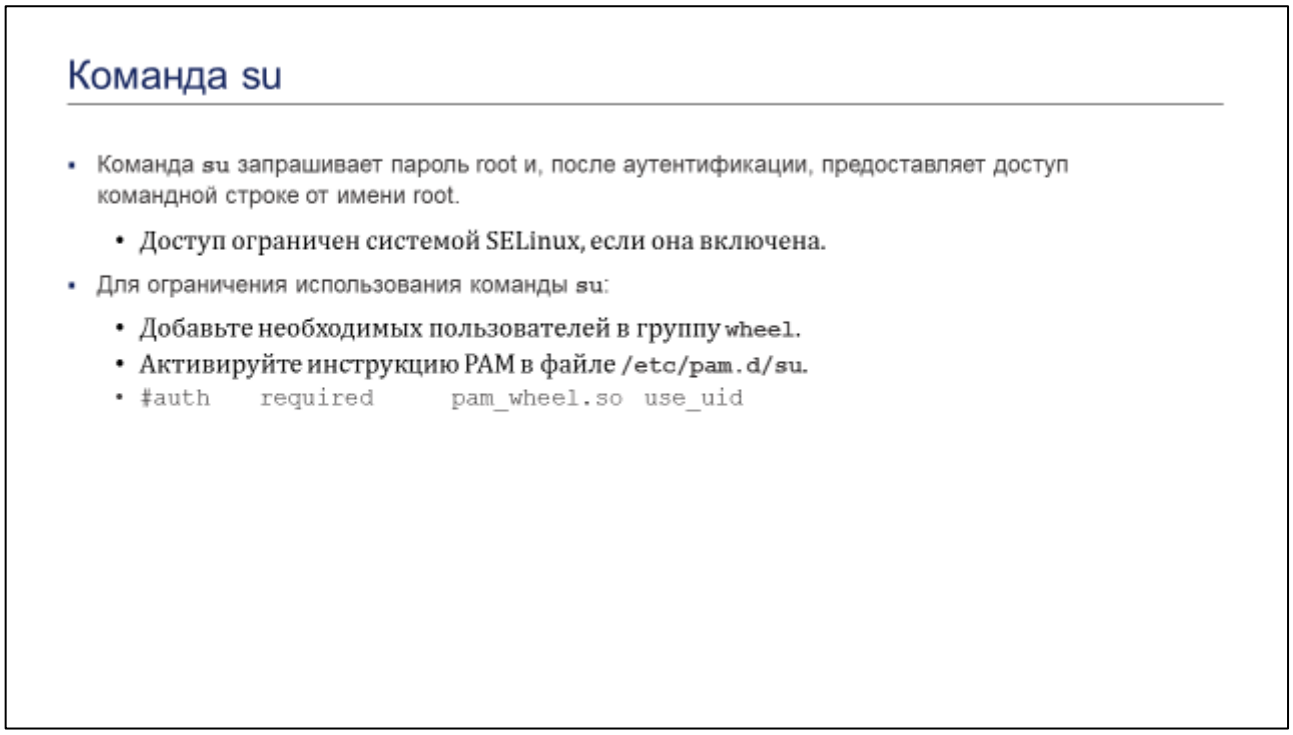
- **pwunconv(8)** – страница руководства для команды **pwunconv**, содержит информацию о конвертации теневых паролей в стандартные.

Для получения информации о связанных конфигурационных файлах, воспользуйтесь следующими страницами руководства:

- **group(5)** – страница руководства для файла **/etc/group**, содержит информацию об определении групп в системе.
- **passwd(5)** – страница руководства для файла **/etc/passwd**, содержит информацию об определении пользователей в системе.
- **shadow(5)** – страница руководства для файла **/etc/shadow**, содержит информацию об определении пароля и его параметров.



Системные администраторы и, в некоторых случаях, пользователи, должны выполнять определенные задачи с применением административного доступа. Получение доступа к системе с правами пользователя root несет потенциальную опасность и может привести к причинению непреднамеренного вреда системе и данным. Данный раздел описывает пути получения административных привилегий при помощи программ смены идентификатора пользователя, таких как **su** и **sudo**. Данные программы позволяют указанным пользователям выполнят задачи, которые обычно доступны только пользователю root, при этом поддерживают высокий уровень управления и безопасности системы.



Когда пользователь выполняет команду **su**, у него запрашивается пароль root и, после аутентификации, предоставляет доступ командной строке от имени пользователя root.

После входа с использованием команды **su**, пользователь становиться пользователем root и получает абсолютный административный доступ к системе. Примечательно, что доступ ограничен системой SELinux, если она включена. Дополнительно, после того как пользователь повысил свои привилегии до root, он может использовать команду **su** для переключения в любого пользователя на системе без запроса пароля.

Так как данная программа настолько сильна, администраторы компании могут быть заинтересованы в ограничении доступа к этой команде.

Один из простейших путей сделать это, это добавить пользователя в специальную административную группу wheel. Для этого выполните следующую команду от имени пользователя root:

```
# usermod –a –G wheel USERNAME
```

В предыдущей команде замените *USERNAME* на имя пользователя, которого хотите добавить в группу wheel.

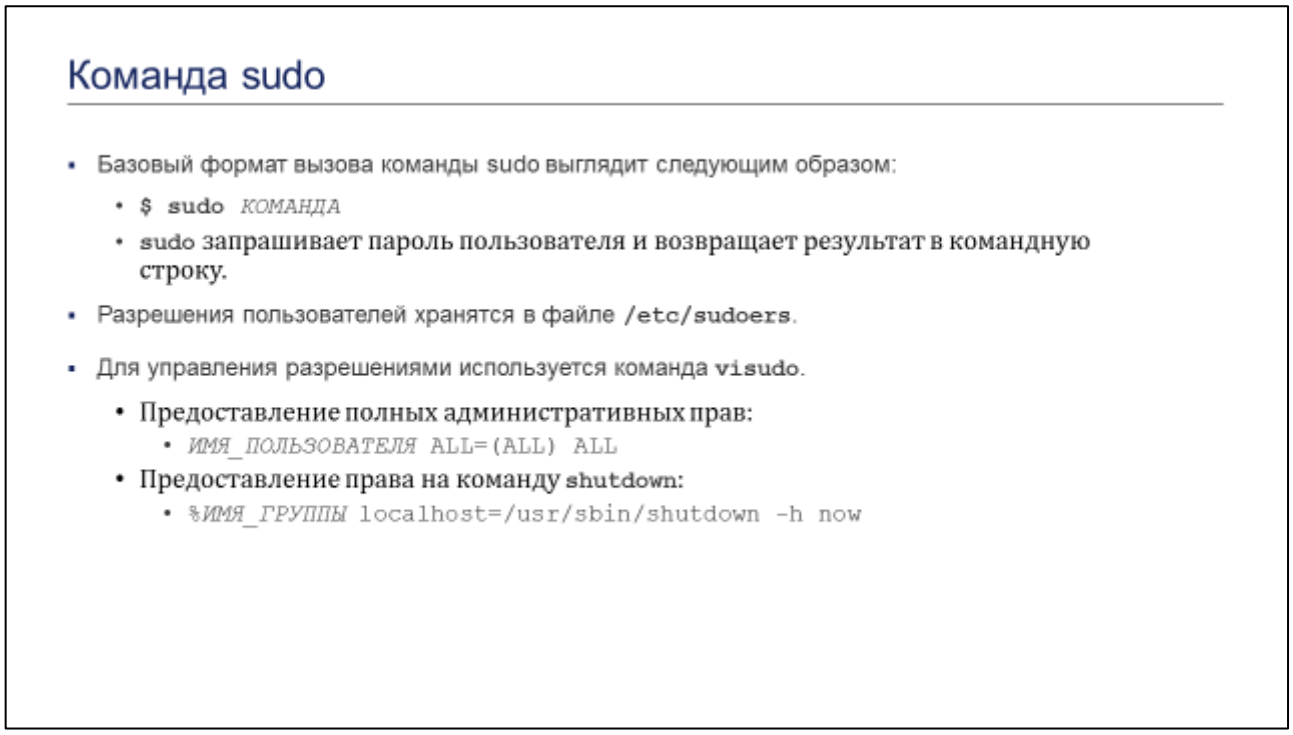
После добавление необходимых пользователей в группу wheel, можно сделать так, чтобы только ее члены смогут использовать команду **su**. Для этого необходимо отредактировать конфигурационный файл Pluggable Authentication Module (PAM) для **su – /etc/pam.d/su**. Откройте файл в текстовом редакторе и раскомментируйте следующую строку, удалив символ **#**:

```
#auth required    pam_wheel.so      use_uid
```

Это изменение обозначает, что только члены административной группы wheel могут переключать пользователя при помощи команды su.

ПРИМЕЧАНИЕ

Пользователь root является частью группы wheel по умолчанию.



Команда **sudo** предлагает иной подход к предоставлению пользователю административного доступа. Когда доверенный пользователь вызывает административную команду при помощи sudo, у него запрашивается его собственный пароль. Затем, после успешной аутентификации и проверки разрешения на выполнение команды, административная команд выполняется с привилегиями пользователя root.

Базовый формат вызова команды **sudo** выглядит следующим образом:

```
$ sudo КОМАНДА
```

В примере выше, команда может быть заменена на команду, зарезервированную для пользователя root, такую как **mount**.

Команда **sudo** обеспечивает высокий уровень гибкости. Для примера, только пользователи, перечисленные в конфигурационном файле **/etc/sudoers** могут пользоваться командой **sudo** и команды выполняются в пользовательской оболочке, не в оболочке пользователя root. Это значит, что оболочка пользователя root может быть полностью отключена.

Каждая успешная аутентификация при помощи команды **sudo** записывается в файл **/var/log/messages**, а вызванная команда вместе с именем вызывавшего записывается в файл **/var/log/secure**. Если требуется расширенное ведение журнала, можно воспользоваться модулем **pam_tty_audit** для включения аудита **TTY** для указанных пользователей при помощи добавления следующих строк в ваш файл **/etc/pam.d/system-auth**:

```
session required pam_tty_audit.so disable=pattern enable=pattern
```

Где *pattern* заменяется разделенным запятыми списком пользователей с опциональным использование подстановочных символов. Например, следующая конфигурация включает аудит **TTY** для пользователя root и отключает его для всех остальных пользователей:

```
session required pam_tty_audit.so disable=* enable=root
```

ПРИМЕЧАНИЕ

Настройка модуля PAM **pam_tty_audit** для аудита **TTY** записывает только ввод **TTY**. Это значит, когда пользователь, на котором ведется аудит, входит в систему, **pam_tty_audit** записывает вводимые пользователем строки в файл **/var/log/audit/audit.log**. Для получения дополнительной информации, обратитесь к странице руководства **pam_tty_audit(8)**.

Другим преимуществом команды sudo является возможность администратора разрешить разным пользователям доступ к указанным командам в соответствии с их нуждами.

Для редактирования конфигурационного файла **/etc/sudoers** администраторы должны использовать команду **visudo**.

Чтобы дать кому-либо полные административные права, введите **visudo** и добавьте строку указания привилегий:

```
student ALL=(ALL) ALL
```

В данном примере пользователь student может использовать **sudo** с любого хоста и выполнять любую команду.

Пример ниже иллюстрирует гранулированные возможности настройки **sudo**:

```
%users localhost=/usr/sbin/shutdown -h now
```

Данный пример указывает, что любой член системной группы users может вызвать команду **/sbin/shutdown –h now** из локальной консоли машины.

Детальное описание опций файла содержатся на странице **sudoers** руководства **man**.

ПРИМЕЧАНИЕ

Есть несколько потенциальных опасностей о которых нужно знать при использовании команды **sudo**. Вы можете избежать их за счет редактирования конфигурационного файла **/etc/sudoers** при помощи команды **visudo**, как описано выше. Если оставить файл **/etc/sudoers** в его состоянии по умолчанию, то это дает каждому пользователю из группы wheel неограниченные права от имени root.

- По умолчанию, **sudo** хранит пароль пользователя в течении пяти минут. Любое последующее использование команды в течении этого периода не запрашивает пароль пользователя. Это может быть использовано злоумышленником, при условии, что пользователь оставит свою рабочую станцию не заблокированной без присмотра. Это поведение может быть изменено за счет добавления следующей строки в файл **/etc/sudoers**:

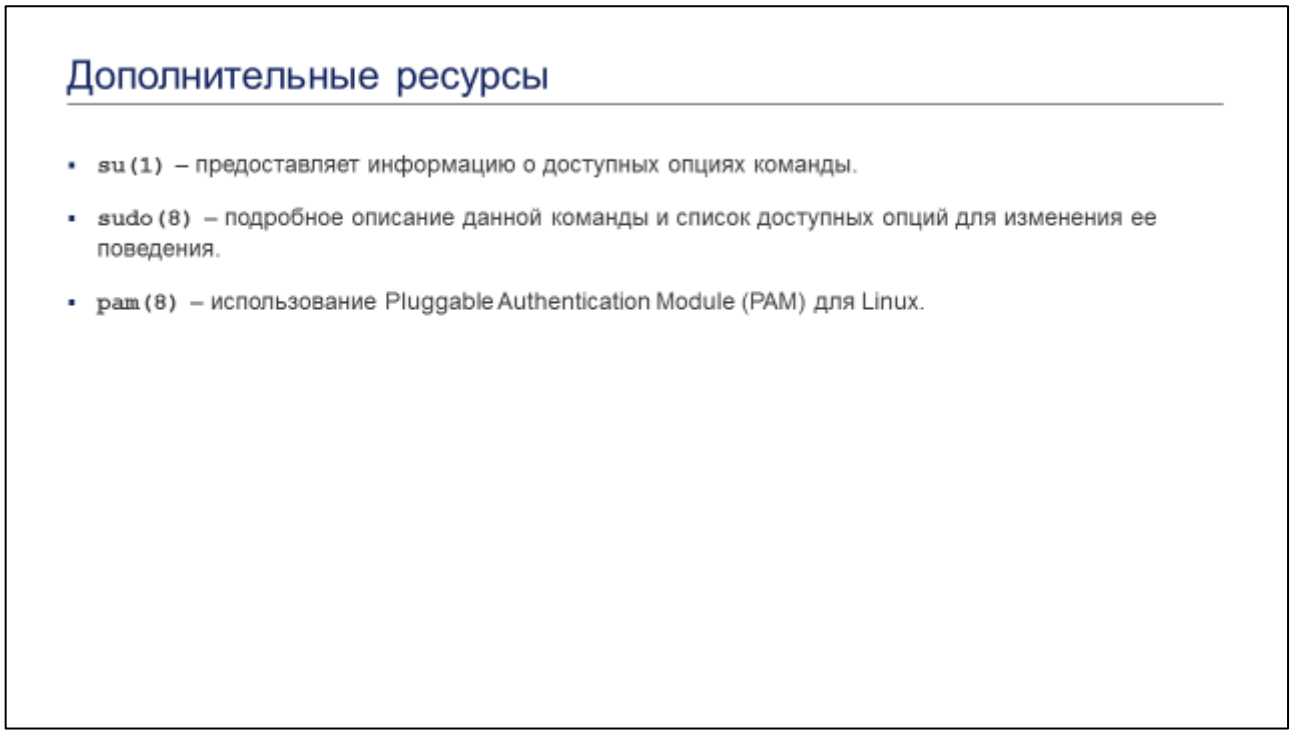
```
Defaults timestamp_timeout=значение
```

Где значение – это разрешенный таймаут в минутах. Установив значение в **0** приведет к запросу пароля при каждом обращении к **sudo**.

- Если учетная запись пользователя **sudo** (**sudoers**) скомпрометирована, злоумышленник может использовать **sudo** для открытия новой оболочки с административными правами:

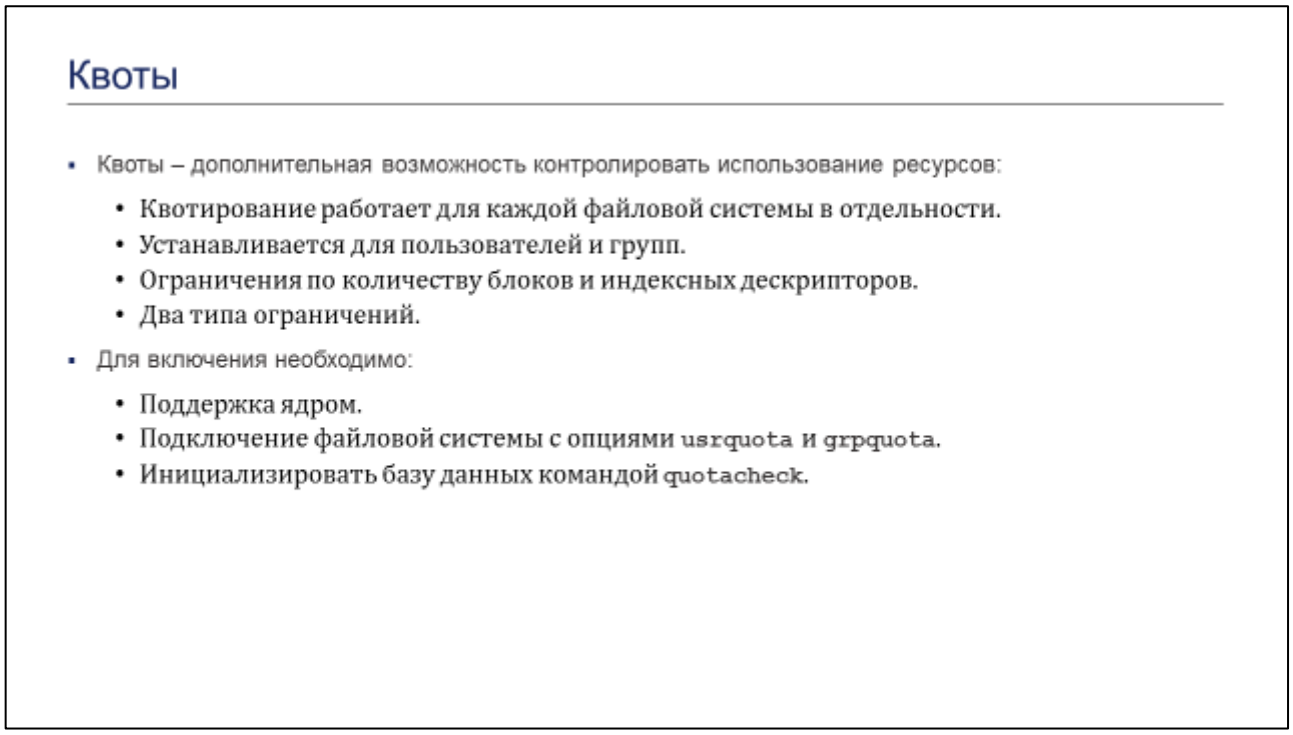
```
$ sudo /bin/bash
```

Открытие новой оболочки от имени пользователя root или аналогичные действия дадут злоумышленнику административный доступ на неограниченное время, превышающее лимиты, прописанные в файле **/etc/sudoers**, а также не будет запрашивать пароль для **sudo** до тех пор, пока открытая сессия не будет закрыта.



Дополнительные ресурсы:

- **su(1)** – страница руководства для **su** предоставляет информацию о доступных опциях команды.
- **sudo(8)** – страница руководства для **sudo** включает подробное описание данной команды и список доступных опций для изменения ее поведения.
- **pam(8)** – страница руководства описывающая использование Pluggable Authentication Module (PAM) для Linux.



Система квотирования позволяет задавать ограничение на использование дискового пространства. Для больших систем, расходование дискового пространства – головная боль, установка квот призвана решить эту проблему.

Чтобы включить квотирование необходимо подключить файловую систему с параметрами **usrquota** – квоты для пользователей, **grpquota** – квоты для групп:

```
# cat /etc/fstab | grep home
/dev/sda2     /home         ext3    defaults,usrquota,grpquota     1 2
```

После включения квот, инициализируйте базу данных командой **quotacheck**:

```
# quotacheck -ug /dev/sda2
```

После создания базу данных, в разделе, для которого было включено квотирование, появятся файлы **aquota.user** и **aquota.group**.

Допускается активация и деактивация системы квотирования без перезагрузки операционной системы, достаточно использование всего двух команд – **quotaon** и **qoutaoff**, которые включают и выключают систему квотирования:

```
# quotaoff /dev/sdaX
# quotaon /dev/sdaX
```

Когда система квотирования инициализирована, остаётся только назначить квоты пользователям для определённых файловых систем, это можно сделать с помощью двух команд, edquota предлагает задать квоты с помощью текстового редактора, указанного в переменной окружения **$EDITOR**, а setquota принимает значения из командной строки, что удобно, при использовании и командных сценариях:

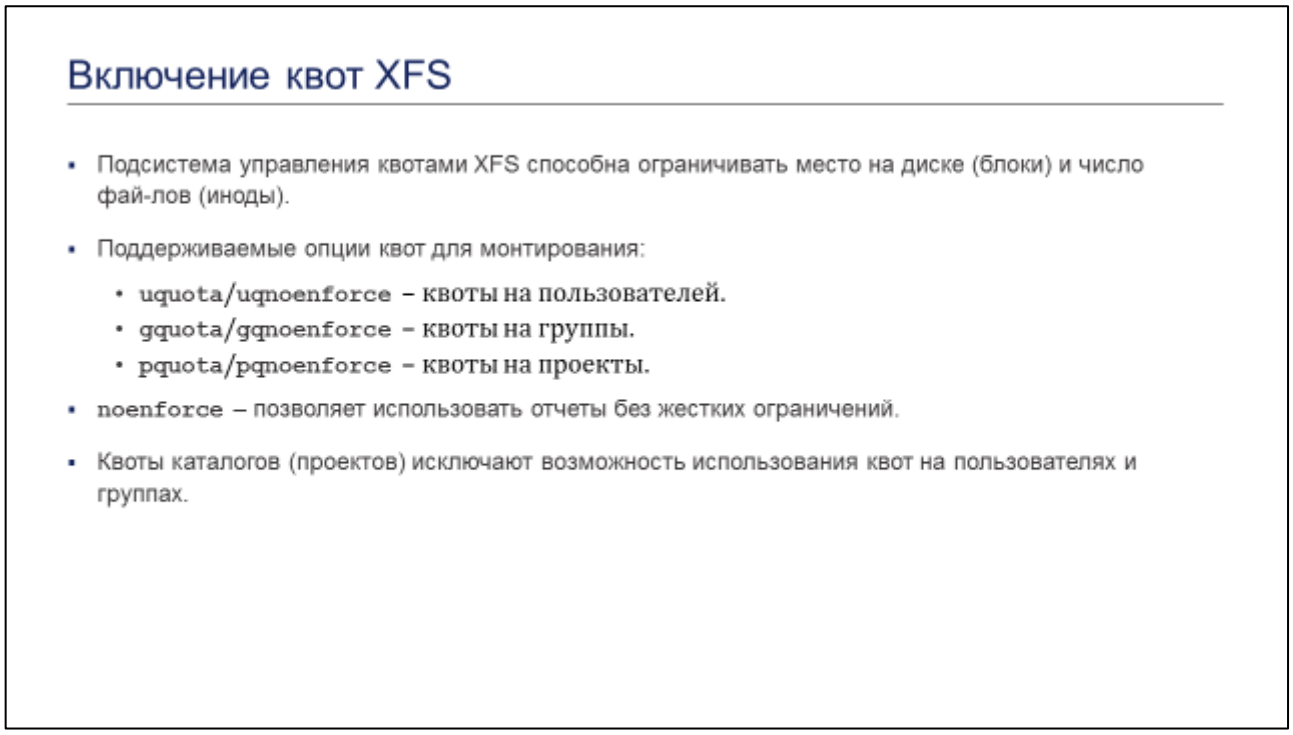
```
# edquota USERNAME
# setquota USERNAME 1024 2048 100 150 /home
```

Обе команды принимают имя пользователя, в качестве аргумента. Программа edquota откроет текстовый редактор, указанный в переменной окружения **$EDITOR** (**vi**, при её отсутствии) с

несколькими колонками. Команда **setquota** сразу принимает необходимые параметры, давайте их разберём:

- **1024** – "мягкая" квота, указывается в блоках. Квоты такого типа обозначают количество блоков, при достижении которого, пользователю будет выдано предупреждение о превышении квоты, а по истечении определённого периода пользователю будет запрещена запись данных на раздел;
- **2048** – "жёсткая" квота, указывается в блоках. При достижении количество блоков, заданного в качестве жёсткой квоты, пользователь больше не сможет записывать данные на раздел;
- **100** – "мягкая" квота применительно к индексным дескрипторам;
- **150** – "жёсткая" квота применительно к индексным дескрипторам.

Чтобы установить срок действия "мягкой квоты", используйте команду: **edquota -t**.



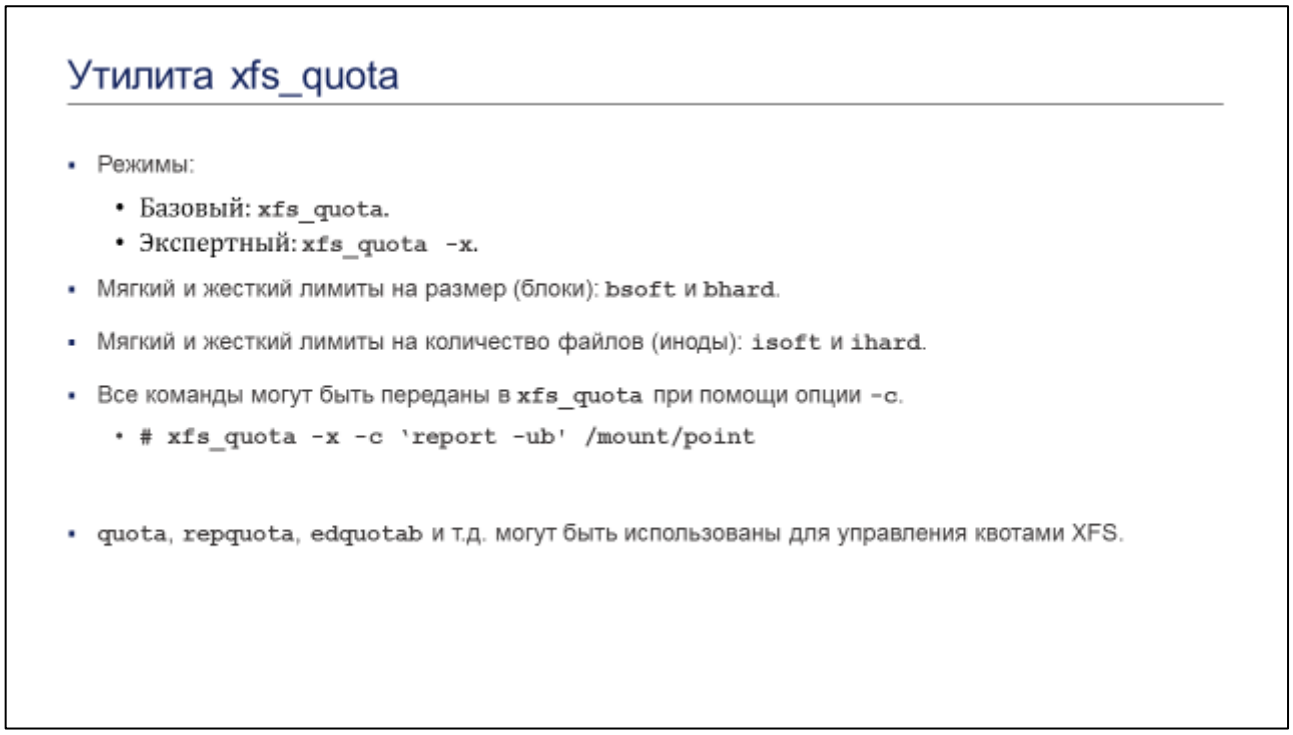
Подсистема управления квотами XFS способна ограничивать место на диске (блоки) и число файлов (иноды). Квоты XFS позволяют управлять и информировать об использовании лимитов пользователями, группами, каталогами или на уровне проектов. Учтите, что квоты на пользователей, группы, каталоги и проекты включаются отдельно друг от друга и квоты каталогов и проектов взаимоисключающие.

При управлении квотами с использованием проектов или каталогов, XFS управляет использованием диска иерархией каталогов, ассоциированной с отдельным проектом. При этом, XFS позволяет кросс-организационную группу связей между проектами. Это принципиально новый уровень, нежели управление квотами на уровне пользователей и групп.

Квоты XFS включаются во время монтирования, при помощи специальных опций. Каждая опция может быть указана как noenforce; это позволяет использовать отчеты без жестких ограничений.

Поддерживаемые опции квот для монтирования:

- **uquota/uqnoenforce** –Квоты на пользователей.
- **gquota/gqnoenforce** – Квоты на группы.
- **pquota/pqnoenforce** – Квоты на проекты.



После включения квот, утилита **xfs_quota** может быть использована для установки ограничений и создания отчетов об использовании диска. По умолчанию **xfs_quota** запускается интерактивно в базовом режиме (Basic Mode). Базовый режим использует команды для упрощения создания отчетов об использовании и доступен всем пользователям.

Базовый режим xfs_quotas поддерживает следующие команды:

```
# quota USERNAME/USERID
```

Отображает использование и лимиты для указанного пользователя (*USERNAME*) или идентификатора пользователя (*USERID*).

```
# df
```

Отображает свободное и использованное количество блоков и инод.

В дополнение, **xfs_quota** поддерживает экспертный режим (Expert Mode). Команды этого режима позволяют менять ограничения и строить более детальные отчеты. Экспертный режим доступен только пользователям с повышенными привилегиями. Для интерактивного использования экспертного режима, запустите: **xfs_quota -x**.

Экспертный режим xfs_quotas поддерживает следующие команды:

```
report /path
```

Отчет по квотам для указанной файловой системы.

```
limit
```

Изменение лимитов квот.

Для получения полного списка команд и справки по ним в обоих режимах используйте команду **help**.

Все команды могут быть переданы в **xfs_quota** при помощи опции **-c**, для команд из экспертного режима в паре с **-x**.

Для демонстрации отчета по квотам на каталоге **/home** (устройство: **/dev/***BLOCKDEVICE*), используйте следующую команду:

```
# xfs_quota -x -c 'report -h' /home.
```

Пример вывода:

```
User quota on /home (/dev/blockdevice)
      Blocks
User ID     Used  Soft  Hard  Warn/Grace
---------- ---------------------------------
root       0         0     0    00 [------]
testuser    103.4G         0     0    00 [------]
...
```

Для установки мягкого и жесткого лимита на инод (файлов) для пользователя **john** (Домашний каталог: **/home/john**), используйте следующую команду:

```
# xfs_quota -x -c 'limit isoft=500 ihard=700 john' /home/
```

/home – точка монтирования.

По умолчанию команда **limit** задает квоты для пользователя, для задания квоты для группы используйте ключ **-g**, аналогично для проектов необходимо использовать ключ **-p**.

Мягкий и жесткий лимиты для блоков могут быть заданы при помощи параметров **bsoft** и **bhard** вместо **isoft** и **ihard**.

Для установки мягкой и жесткой квоты на блоки в размере 1000 Мб и 1200 Мб для группы **accounting** на файловую систему **/target/path** – используйте следующую команду:

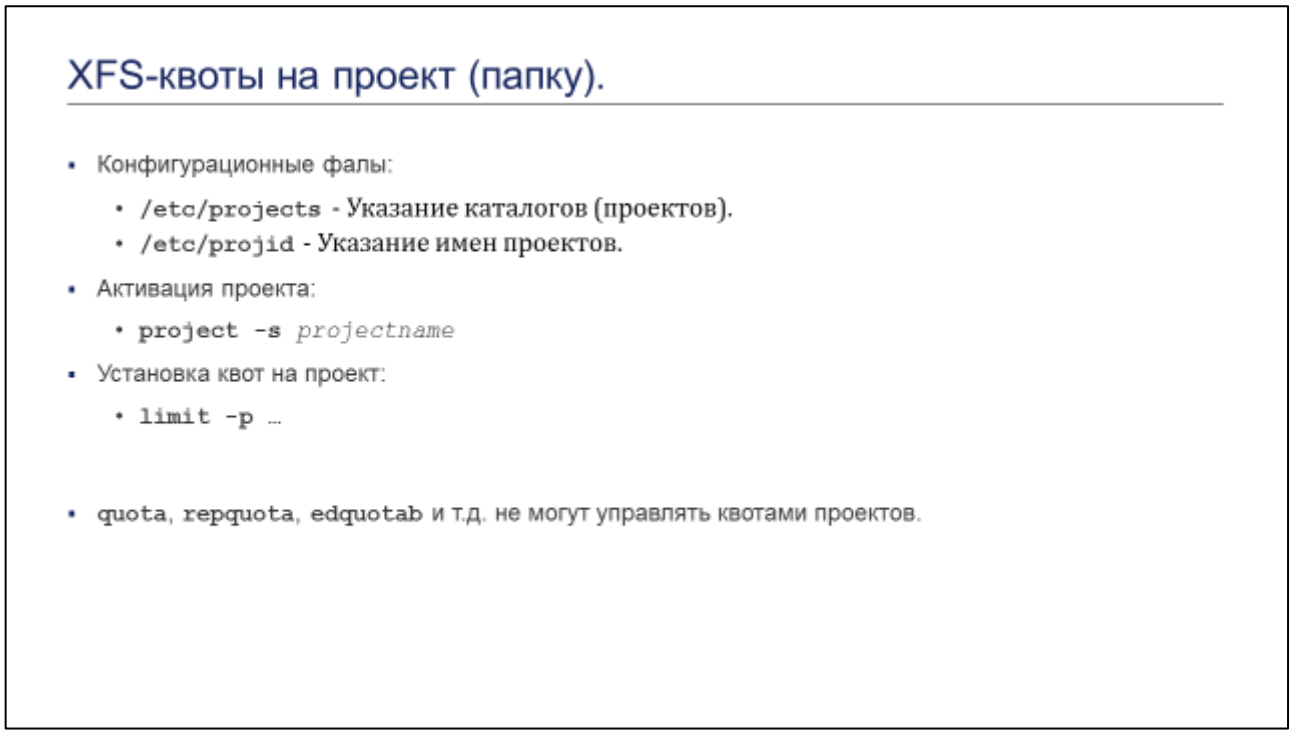
```
# xfs_quota -x -c 'limit -g bsoft=1000m bhard=1200m accounting' /target/path
```

ПРИМЕЧАНИЕ

Параметры **bosft** и **bhard** по умолчанию задаются в байтах.

ПРИМЕЧАНИЕ

Блоки реального времени (**rtbhard**/**rtbsoft**) описаны в **man xfs_quota** как поддерживаемые элементы для задания квоты Субтома реального времени не включены в релиз RedOS 7, поэтому параметры **rtbhard** и **rtbsoft** не применимы.



Установка ограничений на проект.

Перед настройкой ограничений на каталоге при помощи проекта, добавьте их сначала в **/etc/projects**. Имена проектов можно занести в файл **/etc/projected** для связи идентификаторов проектов с именами. После добавления проекта в **/etc/projects**, можно приступать к его инициализации каталога проекта:

```
# xfs_quota -x -c 'project -s projectname' project_path
```

Квоты для проекта с инициализированным каталогом могут быть сконфигурированы при помощи:

```
# xfs_quota -x -c 'limit -p bsoft=1000m bhard=1200m projectname'
```

Классические инструменты управления квотами (**quota**, **repquota**, **edquota**, …) также могут быть использованы для управления квотами XFS, но они не могут быть использованы для управления квотами на проектах.

За дополнительной информацией обратитесь к **man xfs_quota**, **man projid(5)** и **man projects(5)**.

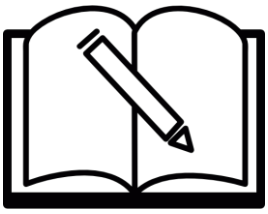
ПРИМЕЧАНИЕ

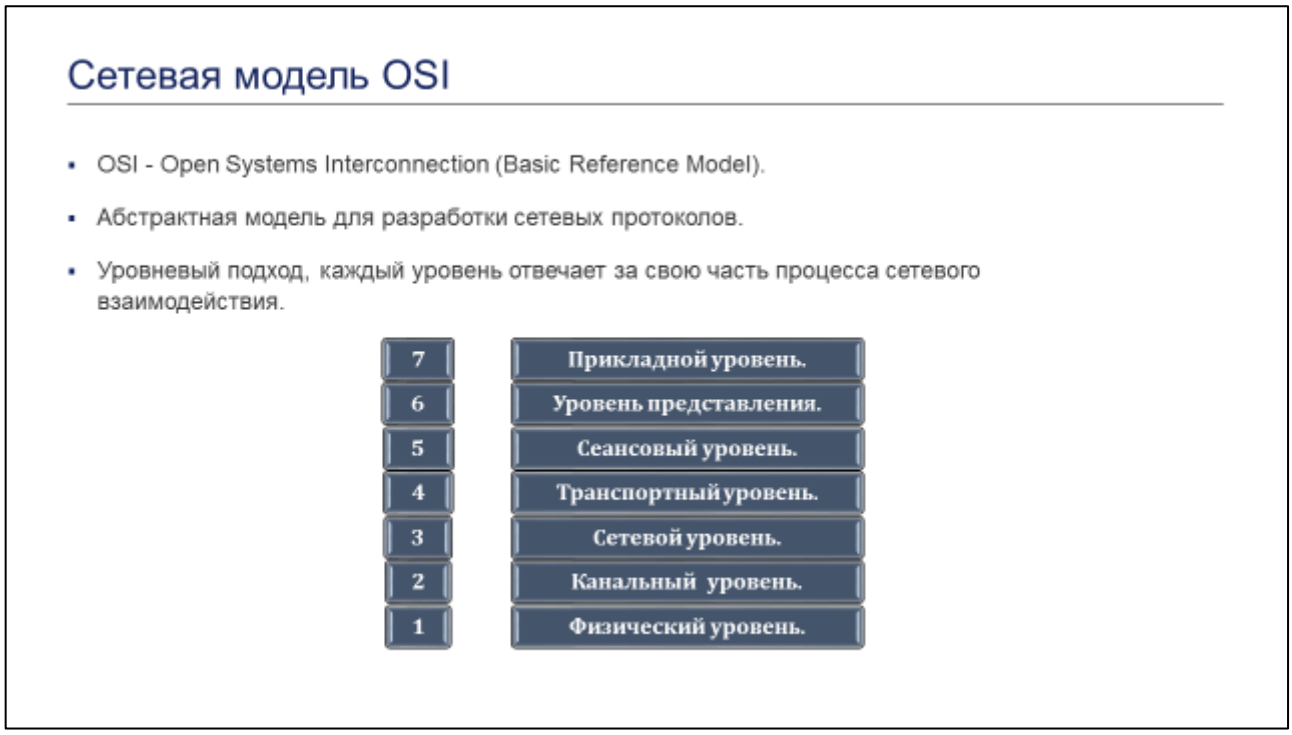
Рекомендуется использовать **xfs_quota**.

Модуль 5. Настройка сетевого взаимодействия в РЕД ОС

В этом модуле рассматриваются:

- Управление сетевыми подключениями
- Демон управления сетью NetworkManager
- Настройка локального разрешения имен





Сетевая модель OSI (Open Systems Interconnection Reference Model-OSI – модель взаимосвязи открытых систем). Эта модель является абстрактной, предлагая уровневую модель построения сети. Такая модель позволяет более структурно подходить к разработке сетевого программного обеспечения. Несмотря на это существует мнение, что модель очень сложна в реализации и является неудачной.

Прикладной уровень (Application Layer) – уровень, который обеспечивает взаимодействие пользователя с сетевой системой. именно на этом уровне работают конечные приложения, предлагающие высокий уровень абстракции. Типичный пример приложения, работающего на 7-м уровне модели OSI – web-браузер. Обеспечивает формирование запросов к 6-му уровню.

Уровень представления (Presentation Layer) – уровень, обеспечивающий кодирование данных. Данные, полученные от приложений, преобразуются в формат для передачи по сети. Данные, полученные из сети, преобразуются в формат для передачи приложениям 7-го уровня.

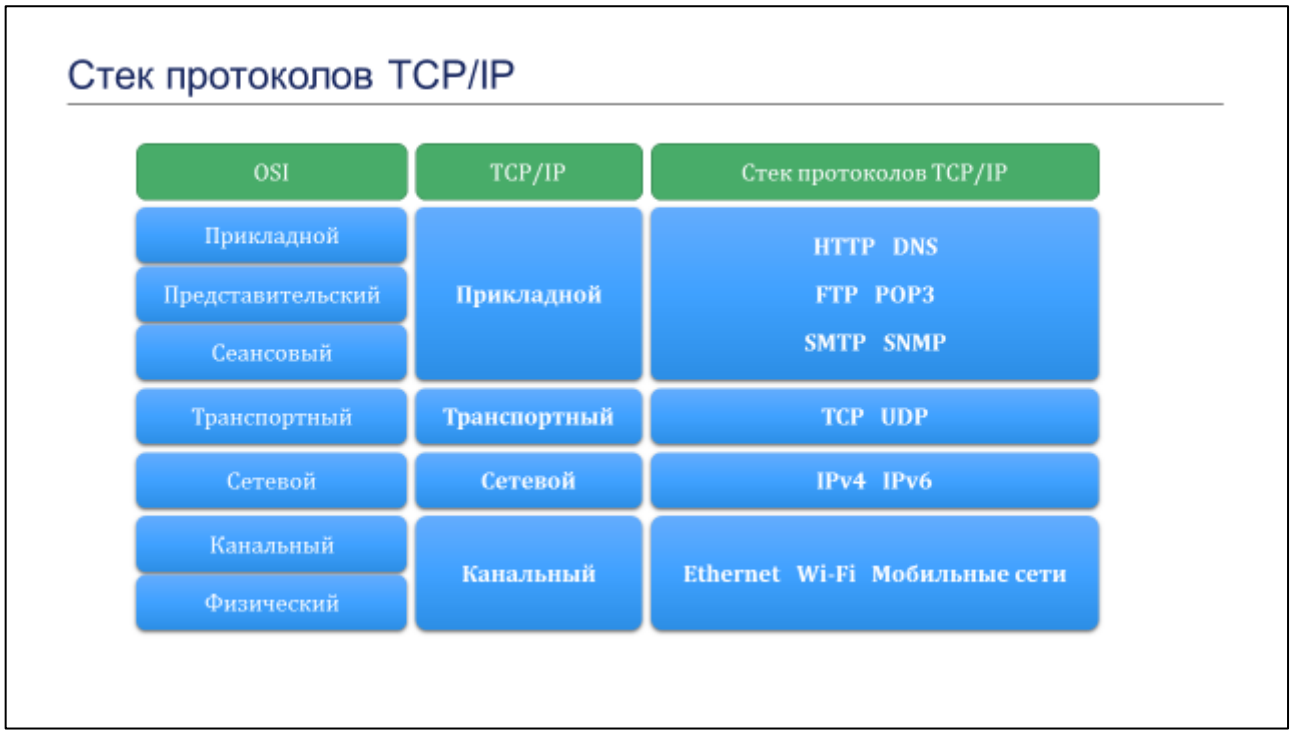
Сеансовый уровень (Session Layer) – 5-й уровень модели OSI обеспечивает поддержание сеанса связи, что позволяет устанавливать соединения на длительное время. Синхронизация обеспечивается с помощью контрольных точек.

Транспортный уровень (Transport Layer) – данный уровень обеспечивает непротиворечивость передаваемых данных. На этом уровне происходит фрагментация данных, определение последовательности и т.д.

Сетевой уровень (Network Layer) – на сетевом уровне модели OSI определяются средства передачи данных. Определение маршрута доставки пакетов, преобразование имён и логических адресов в физические адреса и многое другое.

Канальный уровень (Data Link Layer) – этот уровень является последним программным. Предназначен для обеспечения взаимодействия сетей на физическом уровне. На 2-м уровне обеспечивается определение кадров, проверка их целостности. С точки зрения программного обеспечения, канальный уровень представлен в виде драйвера сетевого устройства.

Физический уровень (Physical Layer) – нижний уровень модели OSI предназначен для непосредственной передачи данных с помощью электрических/оптических сигналов и их приёма.



Стек протоколов TCP/IP – набор сетевых протоколов передачи данных, используемых в сетях, включая сеть Интернет. Название TCP/IP происходит из двух наиважнейших протоколов семейства – Transmission Control Protocol (TCP) и Internet Protocol (IP), которые были разработаны и описаны первыми в данном стандарте.

Протоколы работают друг с другом в стеке (Stack) – это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Например, протокол TCP работает поверх протокола IP.

Стек протоколов TCP/IP включает в себя четыре уровня:

- Прикладной уровень (Application Layer).
- Транспортный уровень (Transport Layer).
- Сетевой уровень (Internet Layer).
- Канальный уровень (Link Layer).

Протоколы этих уровней полностью реализуют функциональные возможности модели OSI. На стеке протоколов TCP/IP построено всё взаимодействие пользователей в IP-сетях. Стек является независимым от физической среды передачи данных.

Протокол IP.

Протокол IP находится не сетевом уровне сетевой модели OSI и является основой семейства протоколов TCP/IP и предназначен для обеспечения межсетевого маршрутизируемого взаимодействия. Основной особенностью является то, что протокол IP не даёт гарантии доставки данных.

На сегодняшний день основное развитие получила четвёртая версия протокола IP (IPV4). Уже в ближайшие годы стоит ожидать повсеместного внедрения протокола версии IPV6, это связано с тем, что при проектировании IPV4 не учли бурного роста сети Internet и адресное пространство стремительно заканчивается.

Протокол TCP.

TCP – один из основных протоколов сети Internet, созданный для управления передачей данных.

Основная особенность протокола TCP – работа с предварительной установкой соединения, что даёт гарантию целостности передаваемых данных. Протокол TCP обеспечивает контроль над процессом передачи информации, осуществление повторов передачи, в случае, когда часть данных была утеряна, исключение дублирования пакетов и т.д. Гарантия целостности данных ведёт за собой повышенное потребление ресурсов, а потому, в случае, когда требуется передавать данные быстро, например, голосовые, используют другие протоколы.

Протокол UDP.

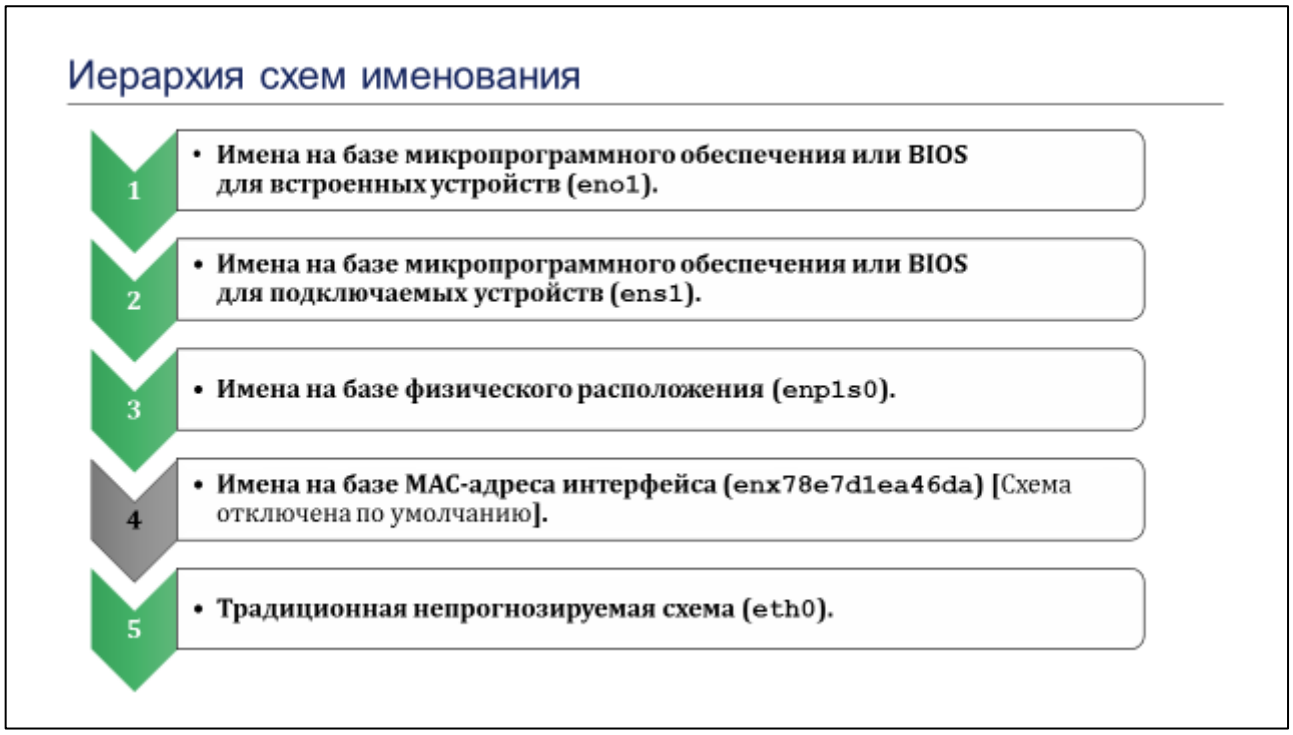
В отличие от протокола TCP, требующего установления соединения, протокол UDP просто отправляет пакеты данных по одному без предварительных договорённостей, более того, удалённый узел не должен подтверждать принятие пакета. В протоколе UDP отсутствует всяческий контроль за правильностью доставки пакета.

Такая особенность делает передачу по протоколу UDP наиболее «дешевой», но и этой дешевизне есть своя цена. В случае, когда всё же нужен контроль за доставкой пакетов, его должно обеспечить само приложение.

Протокол ICMP.

Основная функция протокола ICMP-передача сообщений об ошибках и других исключительных ситуациях, возникающих при сетевом взаимодействии. Дополнительно, ICMP может использоваться для выполнения сервисных функций.

Содержимое ICMP-пакета составляют тип сообщения, код сообщения и само сообщение, зависящее от кода и типа.



Комплексное именование сетевых устройств.

RedOS предоставляет методы для целостного и прогнозируемого именования сетевых интерфейсов для сетевых устройств. Эти возможности меняют имена сетевых интерфейсов на системе для упрощения обнаружения и дифференциации интерфейсов.

Традиционно, сетевые интерфейсы в Linux перечисляются как **`eth[0123…]`**, но эти имена не перекликаются с актуальными разъемами. Современные серверные платформы со множеством сетевых адаптеров могут столкнуться с неопределенными и беспорядочными номерами интерфейсов. Это влияет и на сетевые адаптеры, встроенные в материнские плату (Lan-on-Motherboard или LOM) и на дополнительные адаптеры (с одним или несколькими портами).

В RedOS, **`udev`** поддерживает несколько различных схем именования. По умолчанию назначаются фиксированные имена на базе информации о микропрограммном обеспечении, топологии и расположении. В данном случае преимуществом является полностью автоматическое, полностью прогнозируемое формирование имени, которое остается постоянным, как при добавлении, так и при удалении оборудования (повторного перечисления не происходит), таким образом оборудование может быть заменено без необходимости дополнительного конфигурирования. Недостатком новой системы именования устройств являются непривычные имена, такие как **`enp5s0`**, вместо традиционных **`eth0`** и **`wlan0`**.

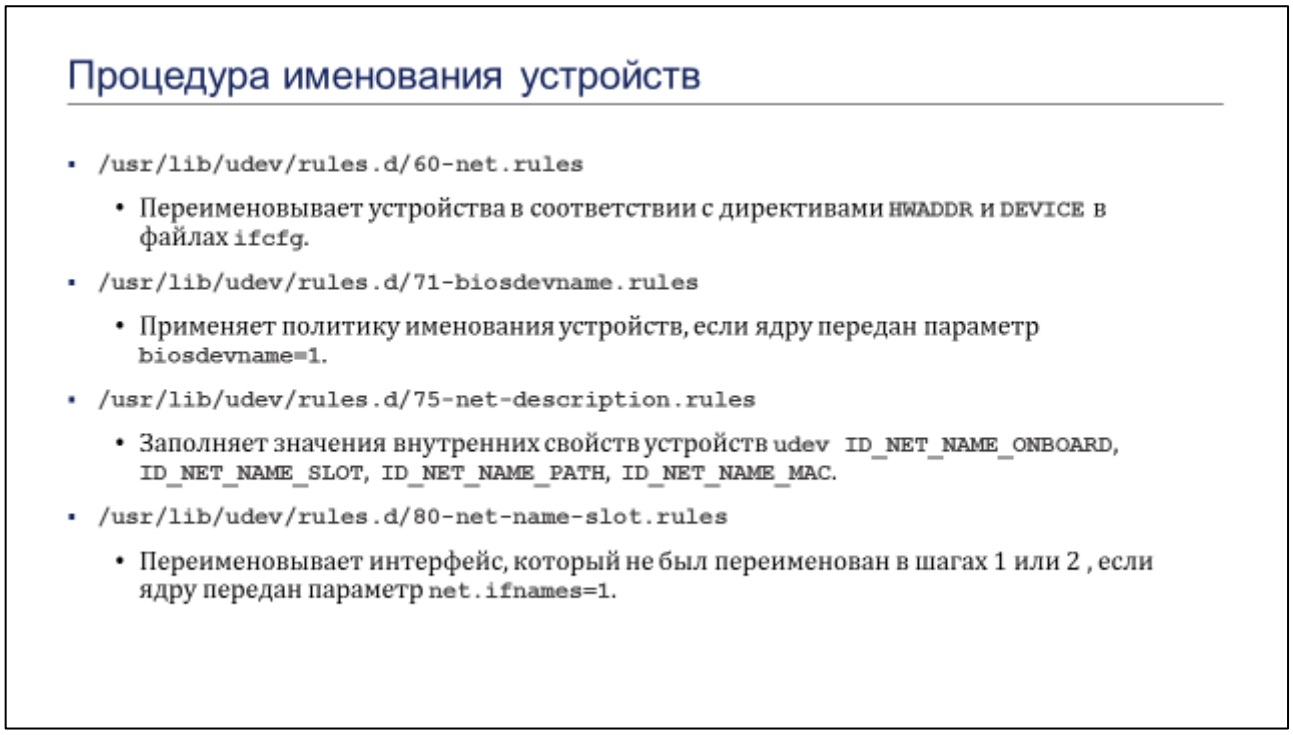
Иерархия схем именования.

По умолчанию systemd именует интерфейсы при помощи следующей политики применения поддерживаемых схем именования:

- **Схема 1**: Имена встроенного микропрограммного обеспечения или BIOS предоставляют номера индекса для встроенных устройств (например, eno1), они применяются при условии, что информация из микропрограммного обеспечения или BIOS доступна и применима, иначе используется схема 2.

- **Схема 2**: Имена встроенного микропрограммного обеспечения или BIOS предоставляют номера индекса слота горячего подключения PCI Express (например, ens1), они применяются при условии, что информация из микропрограммного обеспечения или BIOS доступна и применима, иначе используется схема 3.
- **Схема 3**: Имена встроенного физического расположения соединителя оборудования (например, enp2s0), они применяются если это возможно, иначе используется схема 5.
- **Схема 4**: Имена встроенного MAC-адреса интерфейса (например, enx78e7d1ea46da) по умолчанию не используются, но могут быть выбраны пользователем.
- **Схема 5**: Традиционная непрогнозируемая схема именования используется в том случае, если не один из других методов не применим (например, eth0).

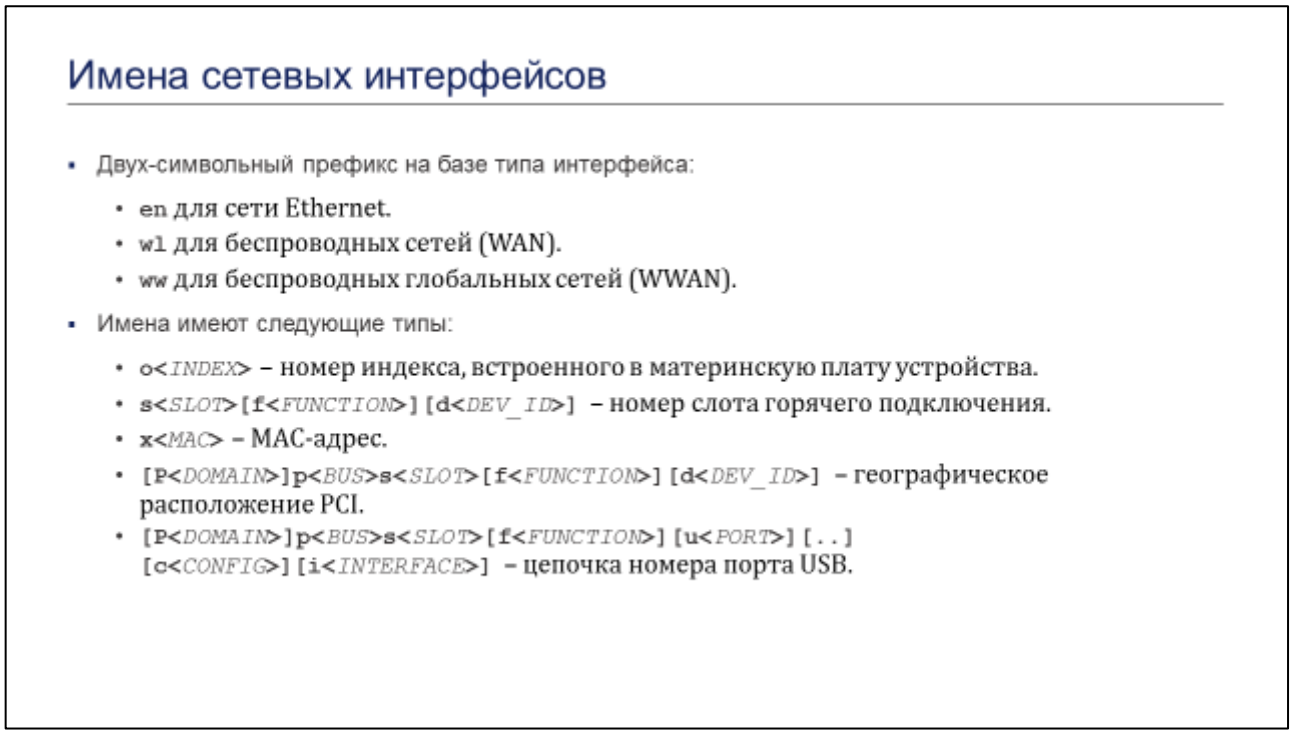
Данная политика, описанная выше, применяется по умолчанию. Если на системе включены biosdevname, то они будут использованы. Для включения **`biosdevname`** необходимо передать ядру параметр **`biosdevname=1`**, исключением являются системы Dell, где **`biosdevname`** используются по умолчанию. При добавлении правил **`udev`**, которые меняют имена устройств ядра, эти правила будут иметь предпочтение при назначении имен устройствам.



Подробная процедура именования выглядит следующим образом:

1. Правило в **/usr/lib/udev/rules.d/60-net.rules** инструктирует вспомогательную утилиту **udev**, **/lib/udev/rename_device** посмотреть во всех файлах **/etc/sysconfig/networkscripts/ifcfg-***suffix*. Если она находит файл **ifcfg** с записью **HWADDR** соответствующей MAC-адресу интерфейса, она переименовывает интерфейс в имя, указанное в директиве **DEVICE**.
2. Правило в **/usr/lib/udev/rules.d/71-biosdevname.rules** инструктирует biosdevname переименовать интерфейс в соответствии с политикой именования устройств, если устройство не было переименовано на предыдущем шаге, **biosdevname** установлен и в процессе загрузки ядру не был передан параметр **biosdevname=0**.
3. Правило в **/lib/udev/rules.d/75-net-description.rules** инструктирует **udev** заполнить значения внутренних свойств устройств **udev ID_NET_NAME_ONBOARD**, **ID_NET_NAME_SLOT**, **ID_NET_NAME_PATH**, I**D_NET_NAME_M**AC, проверяя устройства сетевых интерфейсов. Некоторые свойства устройств могут быть не заполненными.
4. Правила в **/usr/lib/udev/rules.d/80-net-name-slot.rules** инструктируют **udev** переименовать интерфейс, который не был переименован в шагах 1 или 2, а также параметр ядра **net.ifnames=0** не был передан, в таком случае значения будут перебираться в следующем порядке: **ID_NET_NAME_ONBOARD**, **ID_NET_NAME_SLOT**, **ID_NET_NAME_PATH**. Если значение не установлено, проверяется следующее, если ни одно из значений не установлено, интерфейс не переименовывается.

Шаги 3 и 4 применяют схемы именования 1,2, 3 и опционально схему 4. Шаг 2 более детально будет описан далее.

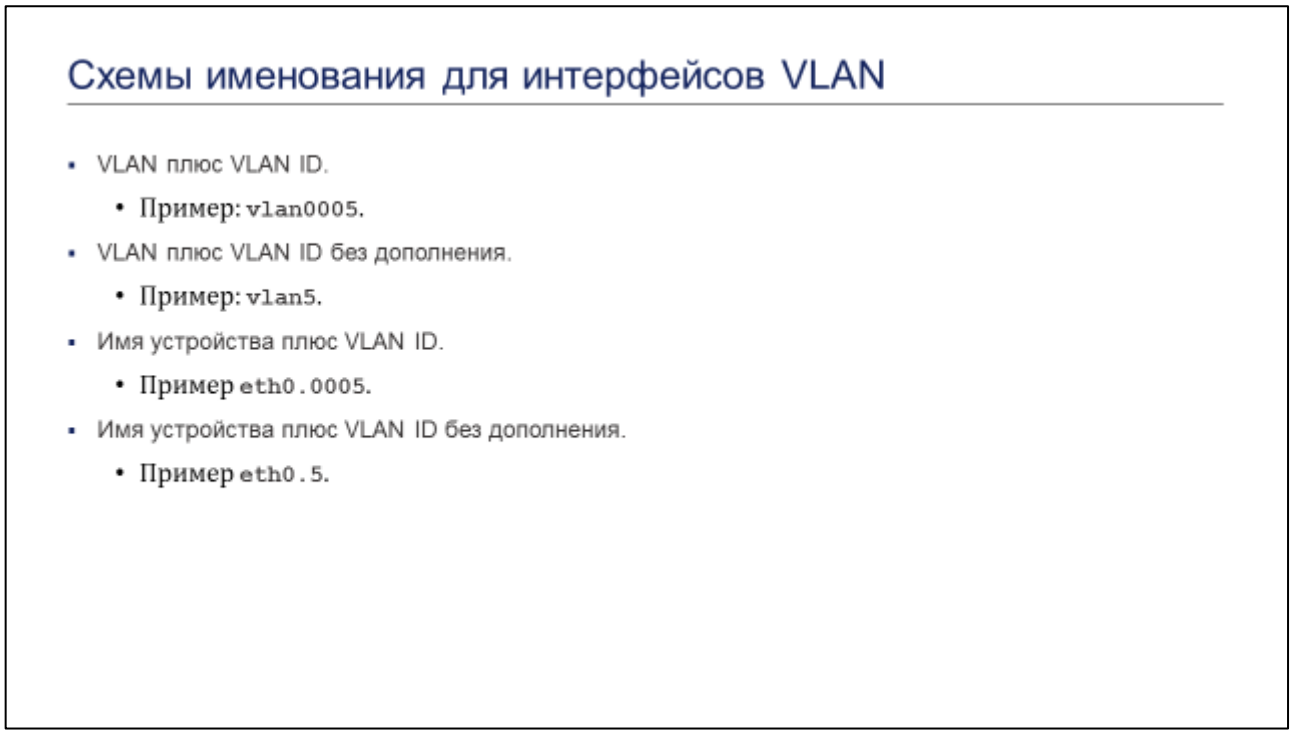


Имена имеют двух-символьный префикс на базе типа интерфейса:

- **en** для сети Ethernet.
- **wl** для беспроводных сетей (WAN).
- **ww** для беспроводных глобальных сетей (WWAN).

Имена имеют следующие типы:

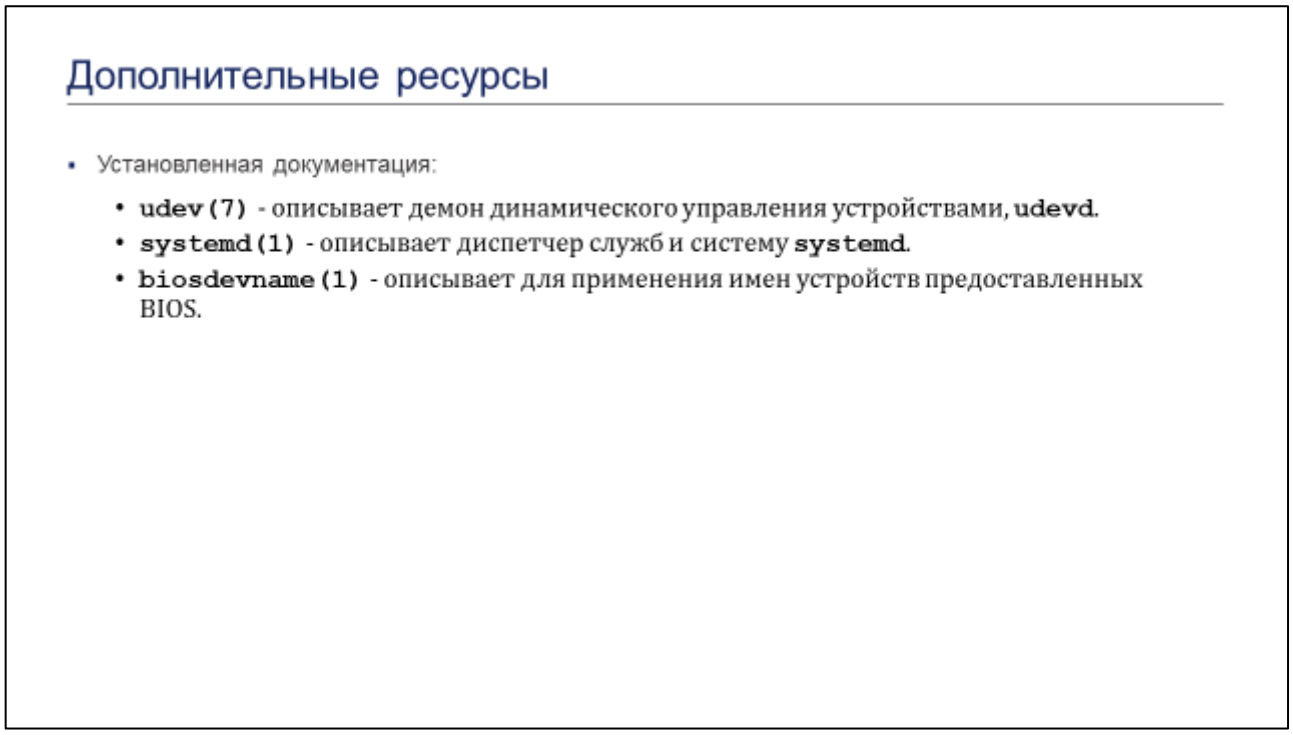
- **o<***index***>** – номер индекса, встроенного в материнскую плату устройства (On-Board Device Index Number).
- **s<***slot***>[f<***function***>][d<***dev_id***>]** – номер слота горячего подключения. Все мультифункциональные PCI-устройства предоставляют номер **[f<***function***>]** в имени устройства, включая устройство с функцией **0**.
- **x<***MAC***>** – MAC-адрес.
- **[P<***domain***>]p<***bus***>s<***slot***>[f<***function***>][d<***dev_id***>]** - географическое расположение PCI. В географическом расположении, номер [**P<***domain***>**] поддерживается только если значение не является **0**. Например: **ID_NET_NAME_PATH=P1enp5s0**.
- **[P<***domain***>]p<***bus***>s<***slot***>[f<***function***>][u<***port***>][..][c<***config***>][i<***interface***>]** – цепочка номера порта USB. Для USB-устройств, полная цепь номеров портов концентраторов объединяются. Если имя получается длиннее максимального значения в 15 символов, имя не экспортируется. Если в цепи множество USB-устройств, предоставляется значение по умолчанию для дескрипторов конфигурации USB (**c1**) и дескриптор интерфейса USB (**i0**).



Традиционно, имена интерфейсов VLAN записываются в следующем формате: имя-интерфейса.VLAN-ID. *VLAN-ID* находится в диапазоне от **0** до **4096**, который максимально состоит из 4 символов и полное имя интерфейса имеет ограничение в 15 символов. Максимальная длина имени интерфейса определяется заголовками ядра и является глобальным ограничением, влияющим на все приложения.

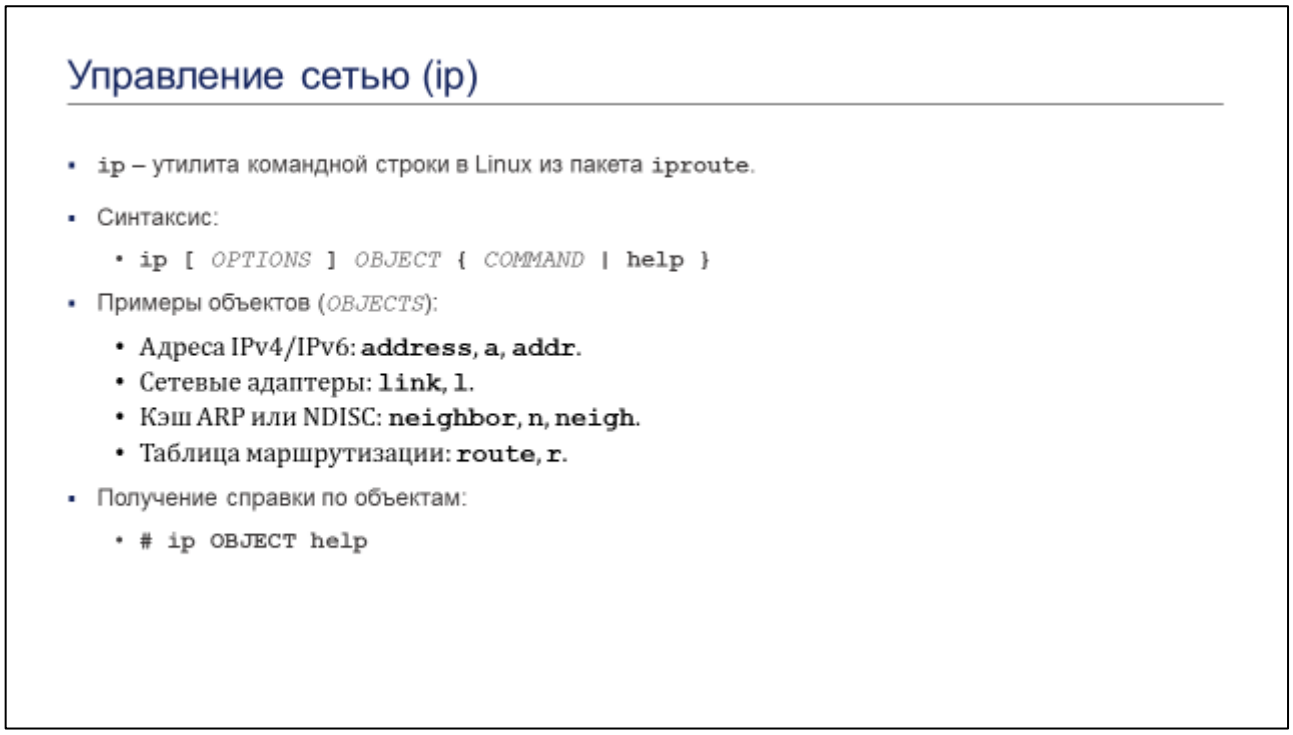
В RedOS 7 для имён интерфейсов VLAN поддерживается четыре соглашения об именовании:

- VLAN плюс VLAN ID – слово **vlan** + VLAN ID. Пример: **vlan0005**.
- VLAN плюс VLAN ID без дополнения – слово **vlan** + VLAN ID без добавления нулей в начале. Пример: **vlan5**.
- Имя устройства плюс VLAN ID – имя родительского устройства плюс VLAN ID. Пример **eth0.0005**.
- Имя устройства плюс VLAN ID без дополнения – имя родительского устройства плюс VLAN ID без добавления нулей в начале. Пример **eth0.5**.



Установленная документация:

- Страница руководства **udev(7)** - описывает демон динамического управления устройствами, udevd.
- Страница руководства **systemd(1)** - описывает диспетчер служб и систему systemd.
- Страница руководства **biosdevname(1)** - описывает для применения имен устройств, предоставленных BIOS.



ip – утилита командной строки в Linux из пакета **iproute**. Позволяет выполнять настройку сетевой подсистемы и является заменой таких утилит, как **ifconfig**, **route**, **arp**.

Синтаксис:

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

Список объектов команды ip, за полным списком обратитесь к документации:

Объект	Аббревиатура	Команды	Описание
address	a, addr	add, del, show, flush	Адреса IPv4/IPv6.
link	l	set, show	Сетевые адаптеры.
neighbor	n, neigh	add, del, change, replace, show, flush	Кэш ARP или NDISC
route	r	list, add, del, flush	Таблица маршрутизации

Получение справки по объектам:

```
# ip address help
# ip r help
```

Примеры использования команды ip.

Отображение информации по всем сетевым интерфейсам:

```
# ip a
```

Отображение только IPv4:

```
# ip -4 a
```

Отображение только IPv6:

```
# ip -6 a
```

Отображение информации по выбранным интерфесам:

```
# ip a show eth0
# ip a list eth0
# ip a show dev eth0
```

Назначение IP-адреса на интерфейс:

```
# ip a add 192.168.0.1/255.255.255.0 dev eth0
# ip a add 192.168.0.1/24 dev eth0
```

Удаление IP-адреса с интерфеса:

```
# ip a del 192.168.1.200/255.255.255.0 dev eth0
# ip a del 192.168.1.200/24 dev eth0
```

Включение и отключение интерфейса:

```
# ip link set dev eth0 down
# ip link set dev eth0 up
```

Отображение ARP-кэша:

```
# ip n show
192.168.1.1 dev eth0 lladr 00:15:5d:46:8b:0a STALE
…
```

Последнее поле отображает состояние записи, возможные состояния:

- **STALE** – запись корректная, но будет проверена перед первой передачей.
- **DELAY** – пакет был отправлен и ожидается подтверждение.
- **REACHABLE** – запись корректна.

ПРИМЕЧАНИЕ

В некоторых сценариях устранения неисправностей может пригодиться возможность внесения изменений в записи ARP-кэша:

Добавление записи:

```
ip neigh add 192.168.1.5 lladdr 00:1a:30:38:a8:00 dev eth0
nud perm
```

Удаление записи:

```
ip neigh del 192.168.1.5 dev eth1
```

Изменение состояния записис:

```
ip neigh chg 192.168.1.100 dev eth1 nud reachable
```

Очистка ARP-кэша:

```
ip -s -s n f 192.168.1.5
ip -s -s n flush 192.168.1.5
```

Отображение таблицы маршрутизации:

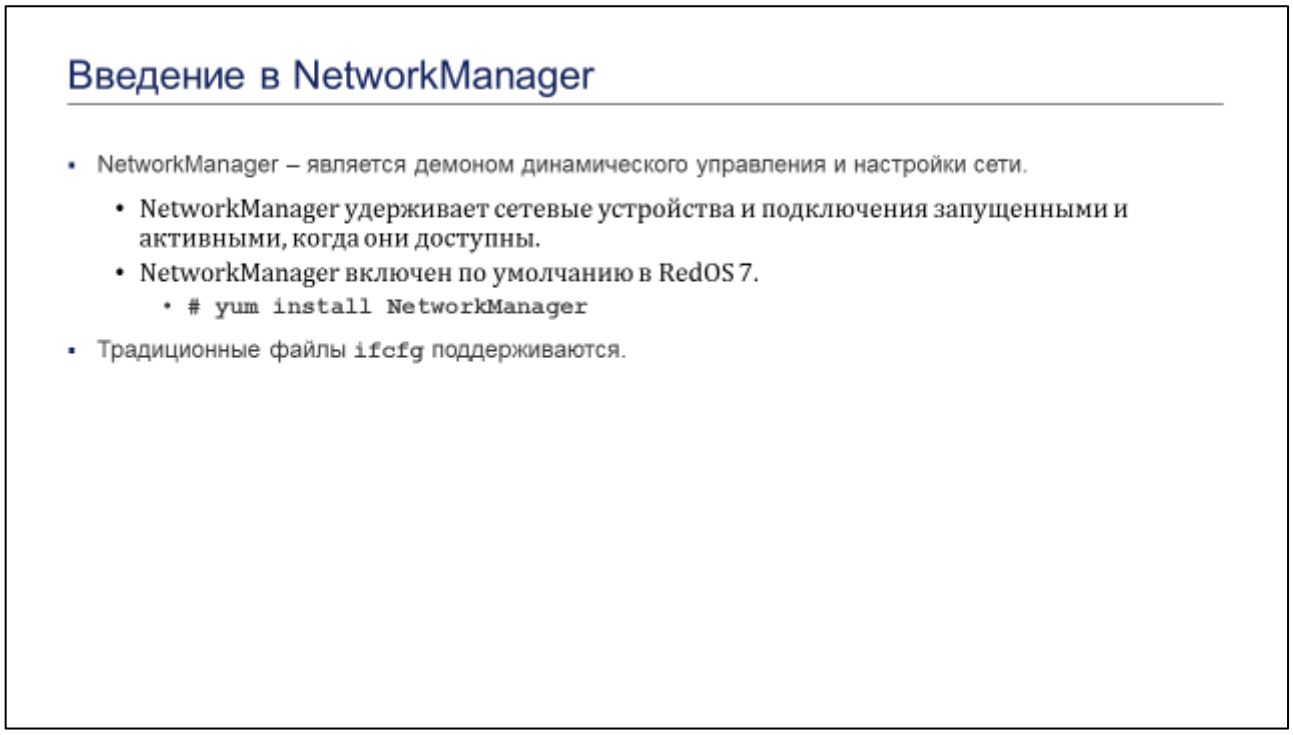
```
# ip r
# ip r list
# ip r list 192.168.10.0/24
```

Добавление нового маршрута:

```
# ip route add 192.168.10.0/24 via 192.168.1.2
```

Удаление маршрута:

```
# ip route del 192.168.10.0/24 dev eth0
```

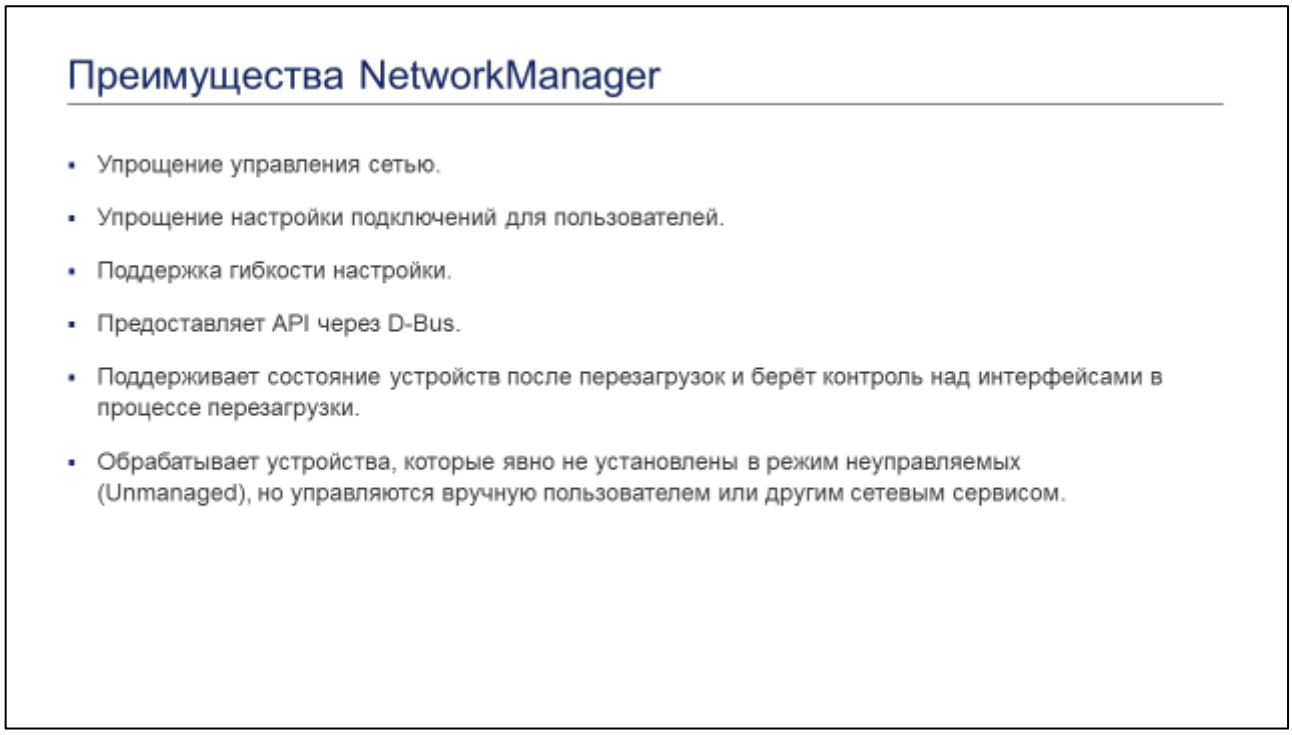


В RedOS 7 служба сети по умолчанию предоставляется NetworkManager, который является демоном динамического управления и настройки сети. NetworkManager удерживает сетевые устройства и подключения запущенными и активными, когда они доступны. Также традиционные файлы **`ifcfg`** по-прежнему поддерживаются.

Установка NetworkManager.

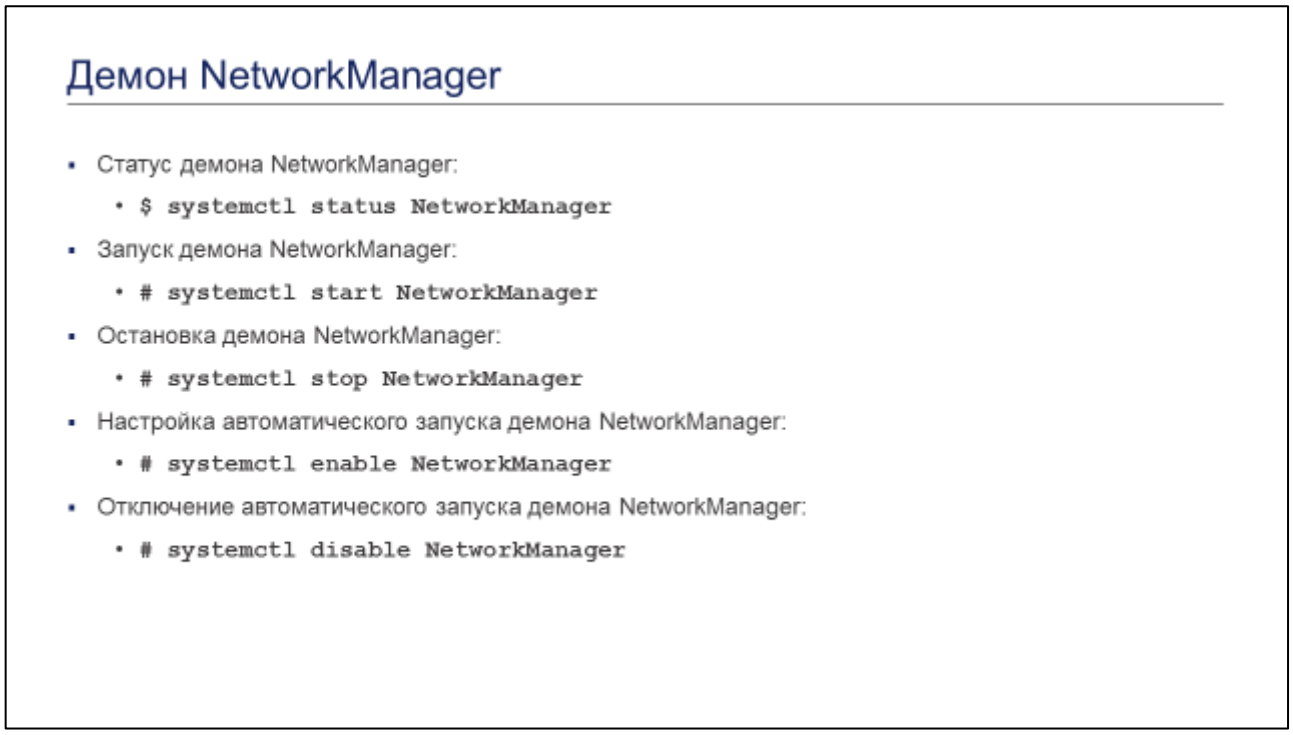
NetworkManager включен по умолчанию в RedOS 7. Для того чтобы убедиться, воспользуйтесь командой:

```
# yum install NetworkManager
```



Основные преимущества от использования NetworkManager:

- Упрощение управления сетью: NetworkManager обеспечивает работу сетевых подключений. Когда он определяет, что на системе отсутствует конфигурация сети, но присутствуют сетевые устройства, NetworkManager создает временные подключения для предоставления сетевого соединения.
- Упрощение настройки подключений для пользователей: NetworkManager предлагает управление через различные инструменты – графические (GUI) и текстовые (nmtui, nmcli).
- Поддержка гибкости настройки: NetworkManager может настраивать псевдонимы, IP-адреса, статические маршруты, информацию DNS и VPN-соединения, а также множество параметров специфических для подключений. Также с его помощью можно изменять настройки в соответствии с потребностями.
- Предоставляет API через D-Bus, который позволяет приложениям запрашивать и управлять настройками и состоянием. Веб-интерфейс Cockpit ведёт мониторинг и обеспечивает возможность выполнять настройки сети через интерфейс D-Bus.
- Поддерживает состояние устройств после перезагрузок и берёт контроль над интерфейсами в процессе перезагрузки, если интерфейс установлен в режим – управляемый (Managed).
- Обрабатывает устройства, которые явно неустановлены в режим неуправляемых (Unmanaged), но управляются вручную пользователем или другим сетевым сервисом.



Демон NetworkManager запускается с правами пользователя root и по умолчанию настроен на автоматический запуск в процессе загрузки. Чтобы проверить статус демона NetworkManager, воспользуйтесь следующей командой:

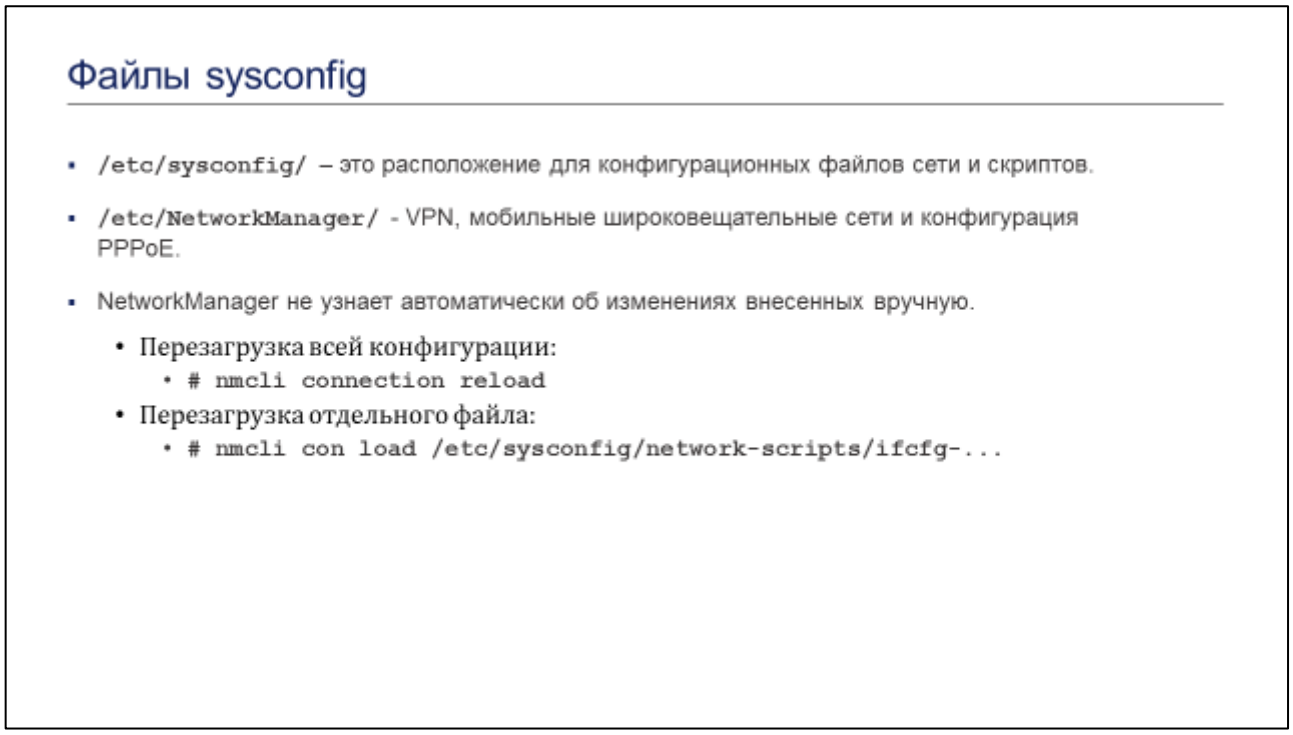
```
$ systemctl status NetworkManager
```

Если NetworkManager не запущен, воспользуйтесь следующей командой:

```
# systemctl start NetworkManager
```

Для настройки автоматического запуска NetworkManager в процессе загрузки системы, воспользуйтесь следующей командой:

```
# systemctl enable NetworkManager
```



Каталог **/etc/sysconfig/** – это расположение для конфигурационных файлов и скриптов. Большая часть информации о настройке сети хранится здесь, за исключением VPN, мобильных широковещательных сетей и конфигурации PPPoE, которые хранится в подкаталогах **/etc/NetworkManager/**. Например, информация о конкретных интерфейсах хранится в файлах **ifcfg** в каталоге **/etc/sysconfig/network-scripts/**.

Для глобальных параметров используется файл **/etc/sysconfig/network**.

В RedOS 7, NetworkManager автоматический не узнаёт об изменениях, внесённых вручную в файлы **ifcfg** и должен быть уведомлен изменениях. Например, если были внесены изменения при помощи редактора, NetworkManager должен прочитать конфигурационные файлы снова, для этого введите следующую команду от имени пользователя root:

```
# nmcli connection reload
```

В качестве альтернативы, чтобы перезагрузить один изменённый файл **ifcfg-***ifname*, воспользуйтесь следующей командой:

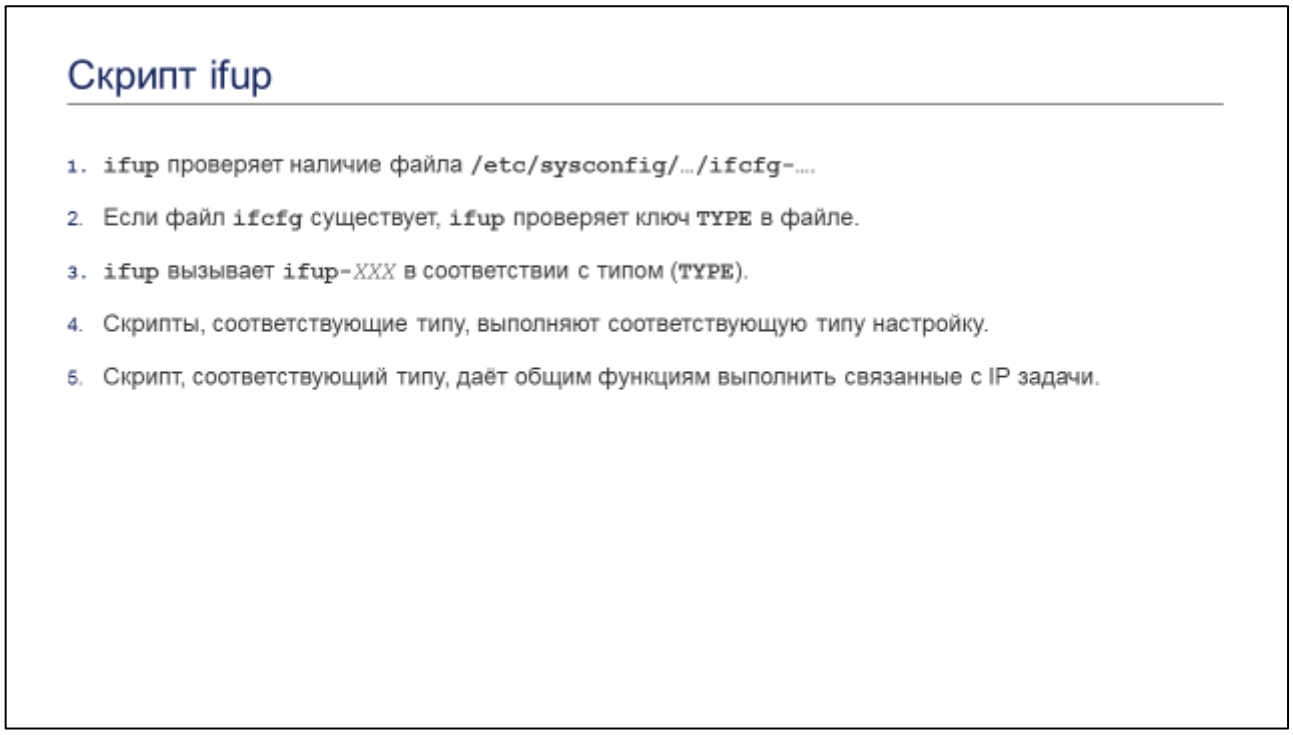
```
# nmcli con load /etc/sysconfig/network-scripts/ifcfg-ifname
```

Также можно указать несколько имён файлов.

Изменения, сделанные при помощи инструментов, таких как **nmcli** не требует перезагрузки, но требуют указания интерфейса, чтобы быть вновь применёными в процессе включения и выключения:

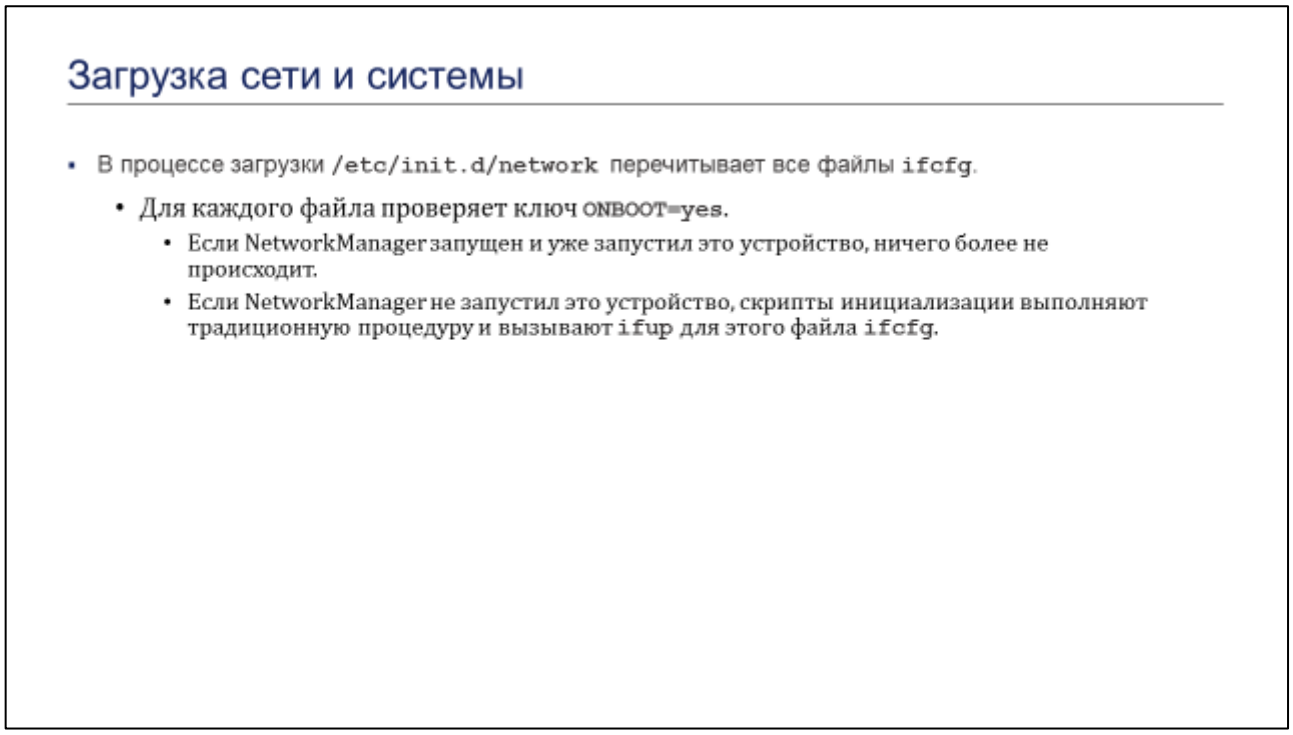
```
# nmcli dev disconnect interface-name
# nmcli con up interface-name
```

NetworkManager не вызывает каких сетевых скриптов, с другой стороны сетевые скрипты могут вызвать NetworkManager если он запущен при использовании команды **ifup**.



Скрипт **ifup** – это общий скрипт, который выполняет несколько задач и затем вызывает скрипты конкретных интерфейсов, такие как **ifup-eth***X*, **ifup-wireless**, **ifup-ppp** и так далее. При запуске **ifup eth0** вручную, происходит следующее:

1. **ifup** проверяет наличие файла **/etc/sysconfig/network-scripts/ifcfg-eth0**.
2. Если файл **ifcfg** существует, ifup проверяет ключ **TYPE** в файле, чтобы определить тип указанного скрипта для вызова.
3. ifup вызывает **ifup-wireless** или **ifup-eth** или **ifup-***XXX* в соответствии с типом (**TYPE**).
4. Скрипты, соответствующие типу, выполняют соответствующую типу настройку.
5. Скрипт, соответствующий типу, даёт общим функциям выполнить связанные с IP задачи, такие как динамическая (DHCP) или статическая конфигурация интерфейса.



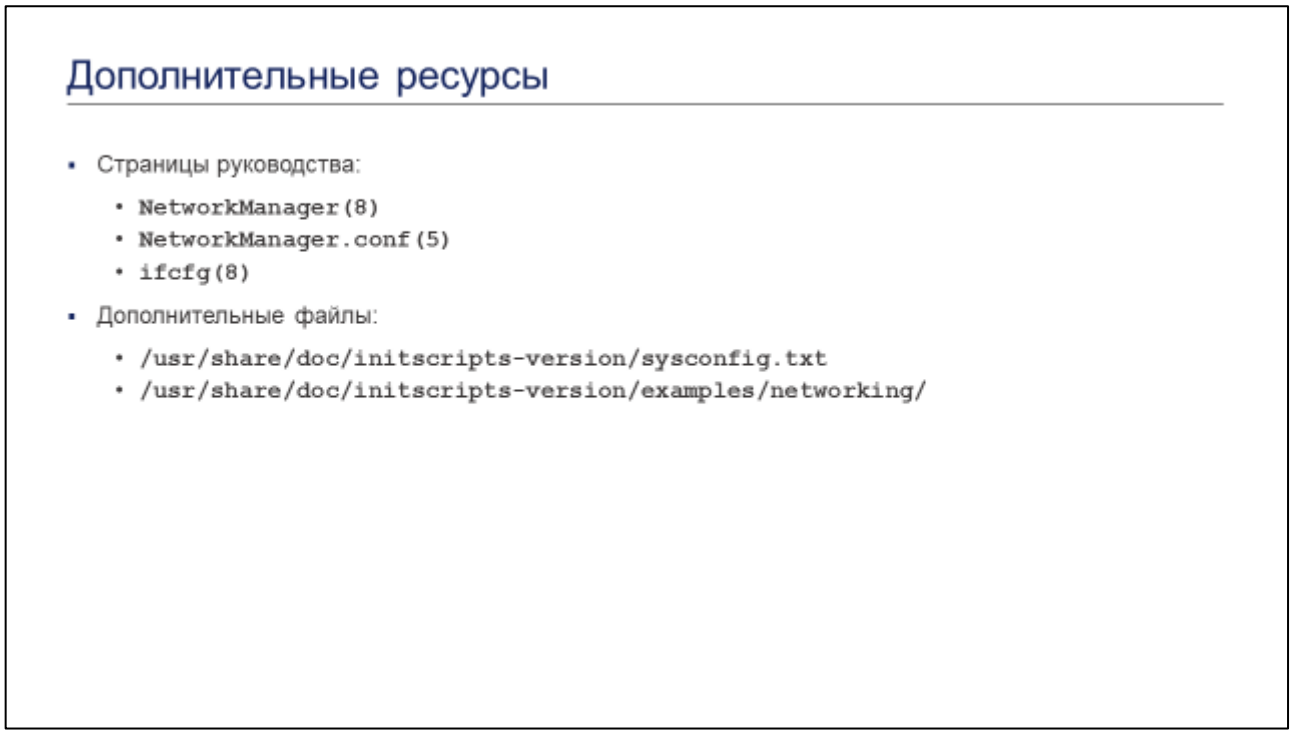
В процессе загрузки **/etc/init.d/network** перечитывает все файлы **ifcfg** и для каждого в котором указан ключ **ONBOOT=yes**, он проверяет, что NetworkManager уже запустил устройство из файла **ifcfg**. Если NetworkManager запущен и уже запустил это устройство, ничего более не происходит, и проверяется следующий файл с ключом **ONBOOT=yes**. Если NetworkManager не запустил это устройство, скрипты инициализации выполняют традиционную процедуру и вызывают **ifup** для этого файла **ifcfg**.

В результате любой файл **ifcfg** в котором указан ключ **ONBOOT=yes** будет запущен в процессе загрузки, при помощи NetworkManager или скриптов инициализации. Это обеспечивает поддержку некоторых типов сетей, которые не обрабатывает NetworkManager, такие как ISDN или аналоговые коммутируемые модемы (Dial-Up), а также настройку сети любыми новыми приложениями, которые не поддерживают NetworkManager на данный момент.

ПРИМЕЧАНИЕ

Не рекомендуется хранить резервную копию файлов где-либо в каталоге **/etc**, или в том же расположение где находятся активные файлы, так как выполняются все скрипты, которые начинаются с **ifcfg-** *. Исключение следующая расширение: **.old**, **.orig**, **.rpmnew**, **.rpmorig**, и **.rpmsave**.

Для получения дополнительной информации по скриптам ifcfg обратитесь к странице руководства **ifcfg(8)**.

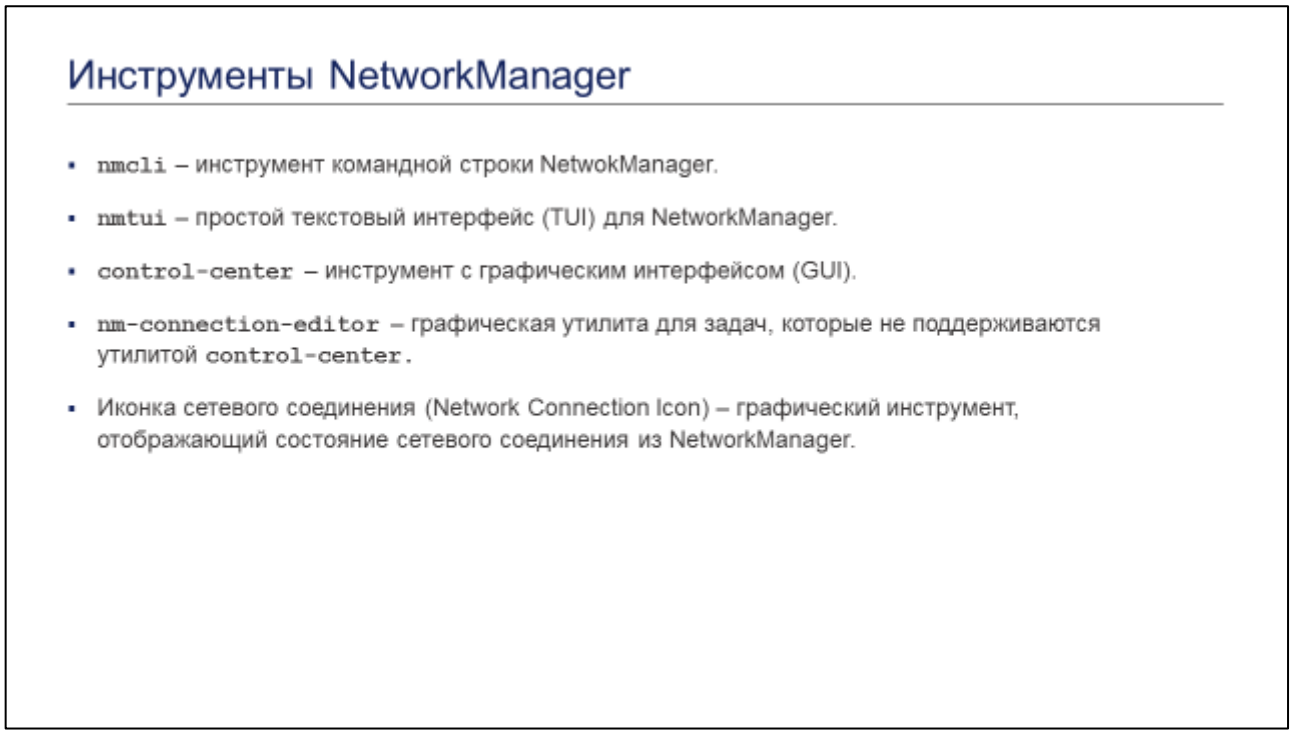


Страницы руководства:

- **NetworkManager(8)** – описывает демона управление сетью.
- **NetworkManager.conf(5)** – описывает конфигурационный файл NetworkManager.
- **ifcfg(8)** - кратко описывает команду **ifcfg**.

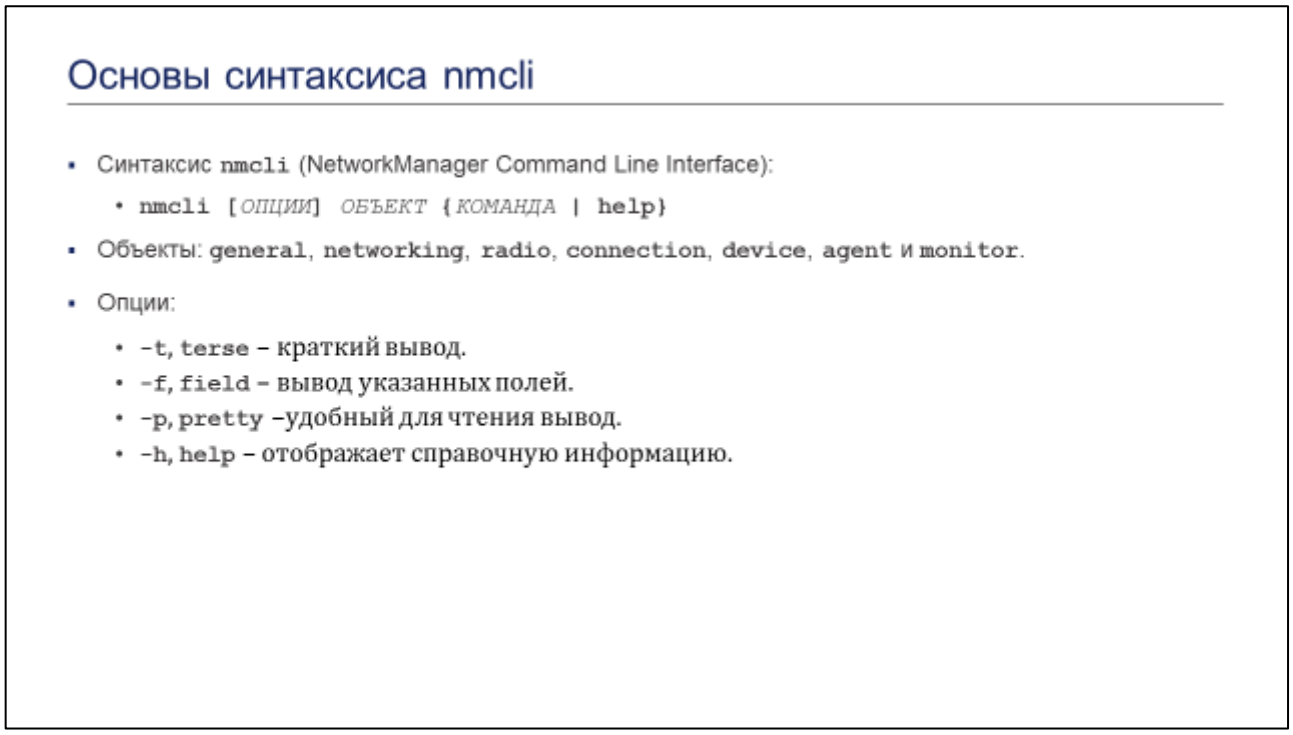
Дополнительные файлы:

- **/usr/share/doc/initscripts-version/sysconfig.txt** - описывает конфигурационные файлы ifcfg и их директивы.
- **/usr/share/doc/initscripts-version/examples/networking/** - каталог с примерами конфигурационных файлов.



Инструменты и приложения NetworkManager для работы с сетью:

- **nmcli** – инструмент командной строки позволяющий пользователям и скриптам взаимодействовать с NetwokManager.
- **nmtui** – простой текстовый интерфейс (TUI) для NetworkManager.
- control-center – инструмент с графическим интерфейсом (GUI), предоставляемый оболочкой.
- **nm-connection-editor** – графическая утилита для определённых задач, которые не поддерживаются утилитой **control-center**, таких как настройка объединений (Bonds) и команд (Teams).
- Иконка сетевого соединения (Network Connection Icon) – графический инструмент, предоставляемый оболочкой Gnome, отображающий состояние сетевого соединения из NetworkManager.



nmcli (NetworkManager Command line Interface) – это утилита командной строки для управления NetworkManager и отображение статуса сети. **nmcli** используется для создания, отображения, редактирования, удаление, активации и деактивации сетевых подключений, а также в качестве средства управления и отображения статусов сетевых устройств.

Утилита **nmcli** может быть использована как пользователями, так и скриптами для управления NetworkManager.

- Для серверов, без дисплейных машин и терминалов, **nmcli** может быть использован для прямого управления NetworkManager, без графического интерфейса пользователя, в том числе для создания, редактирования, запуска и остановки сетевых соединений, а также просмотр статуса сети.
- Для скриптов, **nmcli** поддерживает формат краткого вывода, который лучше подходит обработки в скриптах, это путь интеграции настройки сети, вместо ручного управления сетевыми соединениями.

Базовый формат команды **nmcli** выглядит следующим образом:

```
nmcli [ОПЦИИ] ОБЪЕКТ {КОМАНДА | help}
```

ОБЪЕКТ может быть одним из следующих: **general**, **networking**, **radio**, **connection**, **device**, **agent** и **monitor**. Можно использовать любой префикс этих опций в командах. Например, **nmcli con help**, **nmcli** с **help** и **nmcli connection help** генерируют одинаковый вывод.

Некоторые полезные опции с которых стоит начать:

-t, **terse** – данный режим может быть использован для обработки компьютерной скриптов так как вывод содержит только значения.

```
# nmcli -t device
ens32:ethernet:connected:Wired connection 1
ens33:ethernet:connected:ens33
virbr0:bridge:connected:virbr0
```

```
lo:loopback:unmanaged:
virbr0-nic:tun:unmanaged:
```

-f, **field** – данная опция указывает поля, которые должны быть отображены в выводе. например: **NAME**, **UUID**, **TYPE**, **AUTOCONNECT**, **ACTIVE**, **DEVICE**, **STATE**. Можно использовать одно или несколько полей. При использовании нескольких полей не следует использовать пробел после запятой для разделения полей.

```
# nmcli -f DEVICE,TYPE device
DEVICE      TYPE
ens32       ethernet
ens33       ethernet
virbr0      bridge
lo          loopback
virbr0-nic  tun
```

Или в варианте для скриптов:

```
# nmcli -t -f DEVICE,TYPE device
ens32:ethernet
ens33:ethernet
virbr0:bridge
lo:loopback
virbr0-nic:tun
```

-p, **pretty** – данная опция заставляет **nmcli** создавать удобный для чтения вывод.

```
# nmcli -p device
=====================
  Status of devices
=====================
DEVICE      TYPE      STATE      CONNECTION
-------------------------------------------------------------
ens32       ethernet  connected  Wired connection 1
ens33       ethernet  connected  ens33
virbr0      bridge    connected  virbr0
lo          loopback  unmanaged  --
virbr0-nic  tun       unmanaged  --
```

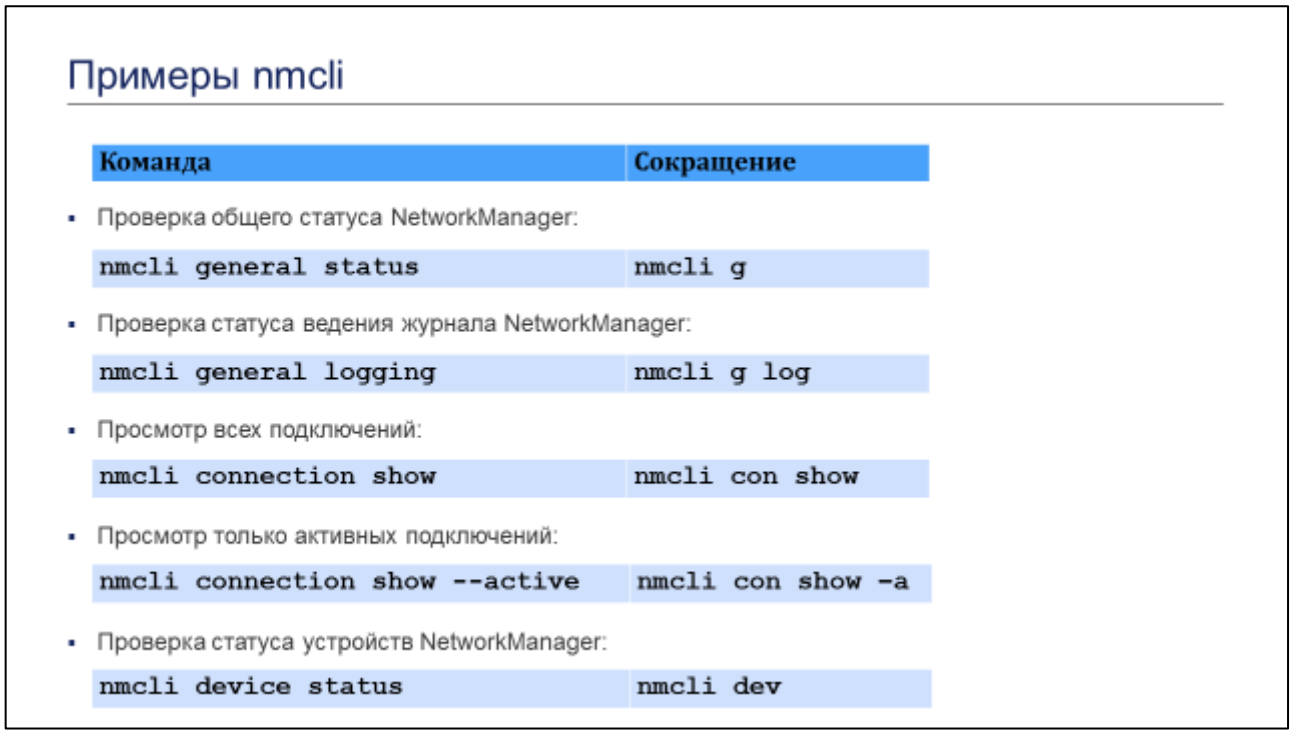
-h, **help** – отображает справочную информацию.

Инструмент **nmcli** обладает встроенный чувствительный к контексту справочной системой.

nmcli help - данная команда выводит список доступных опций и имён объектов для использования в соответствующих командах.

nmcli *ОБЪЕКТ* **help** – данная команда отображает список доступных действий, связанных с указанным объектом.

```
# nmcli c help
```



Пример проверки общего статуса NetworkManager:

```
# nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

В кратком режиме:

```
# nmcli -t -f STATE general
connected
```

Пример просмотра статуса ведения журнала NetworkManager:

```
# nmcli general logging
LEVEL  DOMAINS
INFO
PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,IP4,IP6,AUTOIP4,DNS,VPN,SHARING,S
UPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,INFINIBAND,FIREWALL,ADSL,BOND,V
LAN,BRIDGE,TEAM,CONCHECK,DCB,DISPATCH,AUDIT,SYSTEMD,PROXY
```

Просмотр всех подключений:

```
# nmcli connection show
NAME               UUID                           TYPE       DEVICE
Wired connection 1 7fbc60be-1c41-3c7f-8f84-…  ethernet   ens32
ens33              d744e79d-3e0c-4761-93fe-…  ethernet   ens33
virbr0             0b5a8ed7-9a32-441e-b1dc-…  bridge     virbr0
```

Просмотр только текущих активных подключений:

```
# nmcli connection show --active
NAME                UUID                            TYPE       DEVICE
Wired connection 1  7fbc60be-1c41-3c7f-8f84-…  ethernet   ens32
ens33               d744e79d-3e0c-4761-93fe-…  ethernet   ens33
virbr0              0b5a8ed7-9a32-441e-b1dc-…  bridge     virbr0
```

Просмотр только устройств, распознанных NetworkManager и состояний:

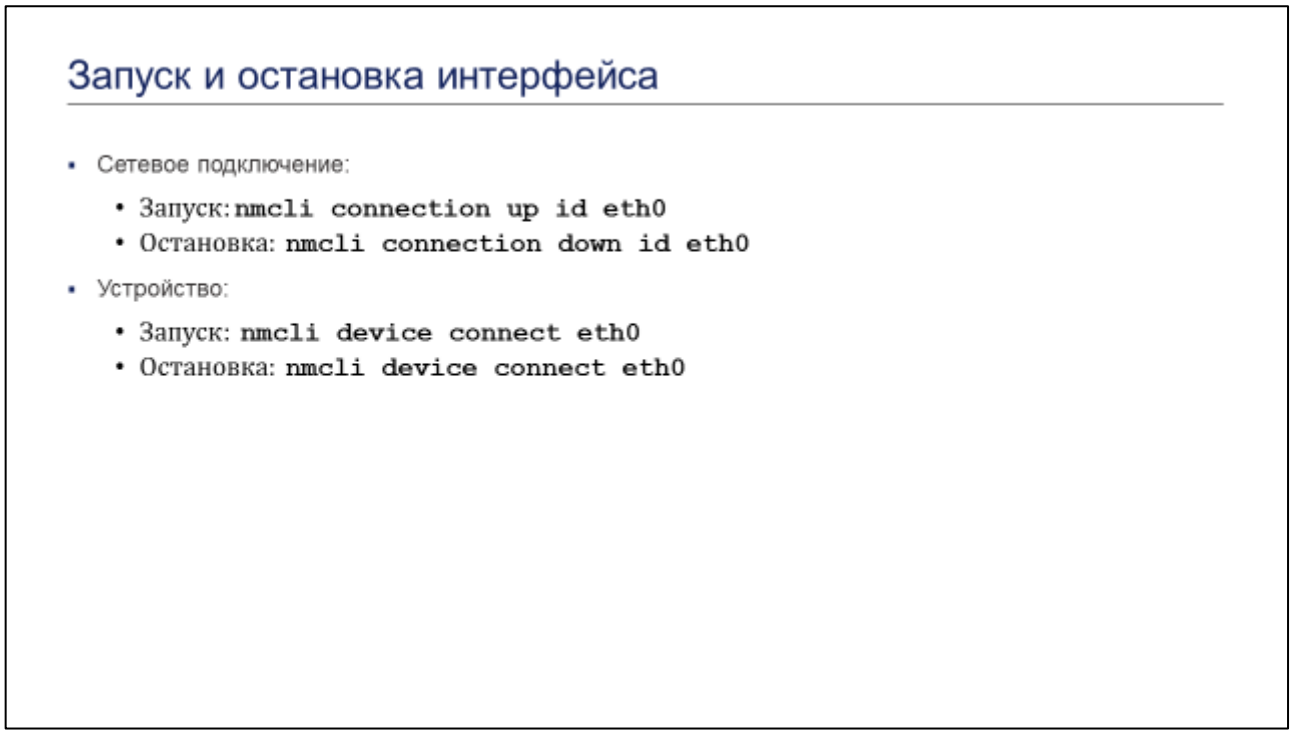
```
# nmcli device status
DEVICE      TYPE      STATE      CONNECTION
```

```
ens32       ethernet  connected  Wired connection 1
ens33       ethernet  connected  ens33
virbr0      bridge    connected  virbr0
lo          loopback  unmanaged  --
virbr0-nic  tun       unmanaged  --
```

Также можно использовать следующие сокращения команды nmcli:

Команда `nmcli`	Сокращение
`nmcli general status`	**`nmcli g`**
`nmcli general logging`	**`nmcli g log`**
`nmcli connection show`	**`nmcli con show`**
`nmcli connection show --active`	**`nmcli con show -a`**
`nmcli device status`	**`nmcli dev`**

Больше примеров доступно на странице руководства **`nmcli-examples(5)`**.

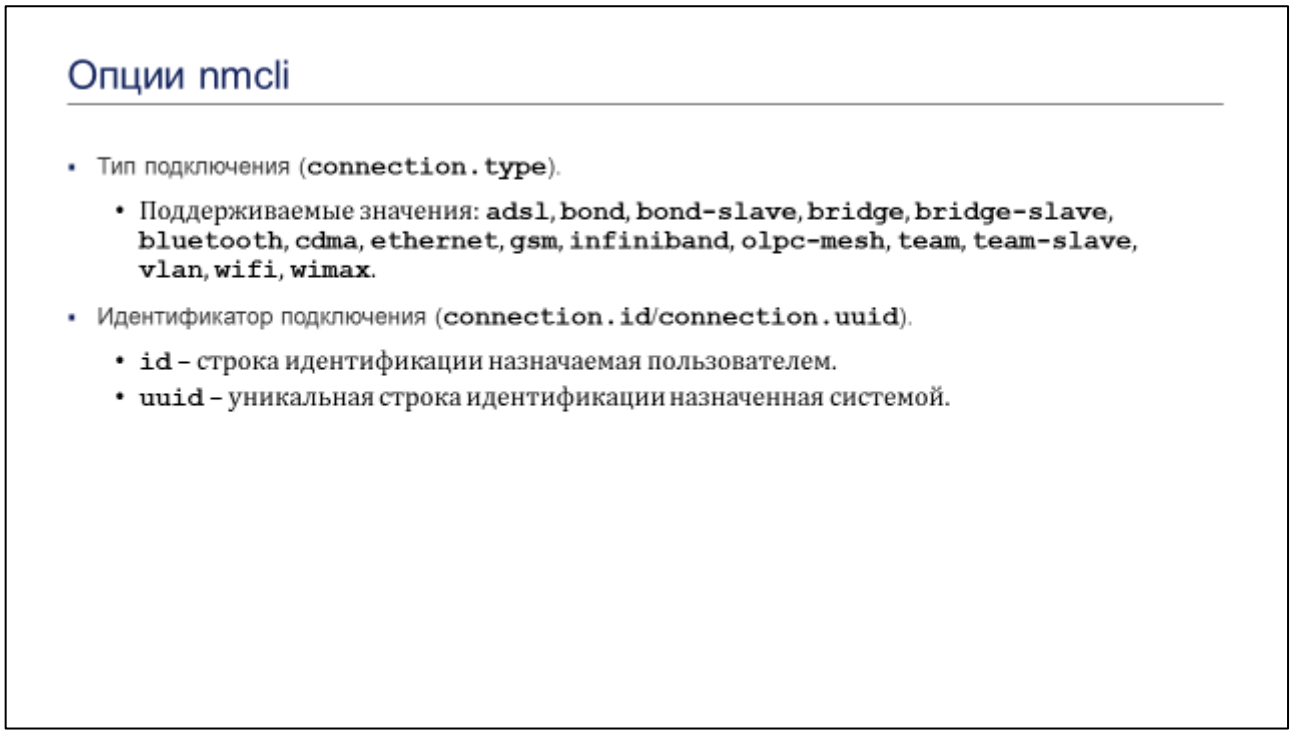


Инструмент **nmcli** может быть использован для запуска и остановки любого сетевого интерфейса в том числе управляющего (Masters).

```
# nmcli con up id bond0
# nmcli con up id port0
# nmcli dev disconnect bond0
# nmcli dev disconnect ens3
```

ПРИМЕЧАНИЕ

Команда **nmcli** connection down деактивируют соединение с устройством не отключая возможность автоматической активации в будущем. Команда **nmcli device disconnect** отключает устройство и не позволяет устройство автоматически активируется в будущем без явного ручного участия.



Далее следуют некоторые свойства опций nmcli. Полный список доступен на странице руководства **nmcli(1)**.

Тип подключения (connection.type).

Позволяет значения: **adsl**, **bond**, **bond-slave**, **bridge**, **bridge-slave**, **bluetooth**, **cdma**, **ethernet**, **gsm**, **infiniband**, **olpc-mesh**, **team**, **team-slave**, **vlan**, **wifi**, **wimax**. Каждый тип подключение имеет специфические для этого типа опции, полный список этих опций(**TYPE_SPECIFIC_OPTIONS**) доступен на странице руководства **nmcli(1)**.

Например, подключение **gsm** требует указания имени точки доступа (Access Point Name Specified) в свойстве **apn**.

```
# nmcli c add connection.type gsm apn ИМЯ_ТОЧКИ_ДОСТУПА
```

Устройство **wifi** требует указания идентификатора сервисного набора (Service Set Identifier) в свойстве **ssid**.

```
# nmcli c add connection.type wifi ssid ИДЕНТИФИКАТОР
```

Имя устройства соответствующего подключению (**connection.interface-name**).

```
# nmcli con add connection.interface-name eth0 type ethernet
```

Имя для профиля подключения (**connection.id**).

Если не указывается имя подключения, то оно генерируется следующим образом:

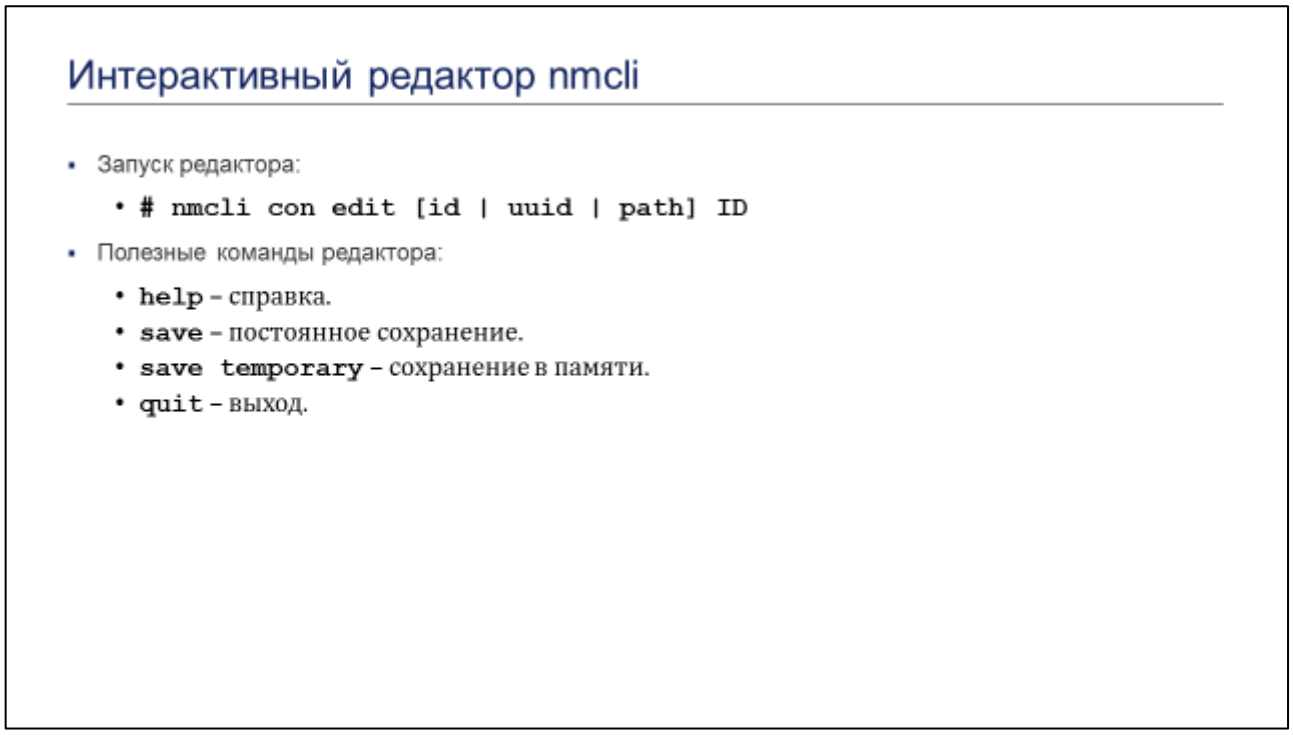
```
connection.type -connection.interface-name
```

connection.id – это имя профиля подключения (Connection Profile) оно не должно соответствовать имени устройства (**wlan0**, **ens3**, **em1**). Однако, пользователи могут именовать подключение после интерфейсов, но это не одно и тоже. На системе может быть несколько профилей подключения доступных для устройства. Обычно это полезно для мобильных устройств или, когда сетевой кабель переключается между различными сетями. Вместо того чтобы

редактировать подключение, создаются разные профили и применяются к интерфейсу по мере необходимости. Опциоя **id** относится к имени профиля подключения.

Наиболее важные опции для команд nmcli, таких как show, up, down:

- **id** – строка идентификации, назначаемая пользователем профилю подключения. Идентификатор может быть использован командах **nmcli connection** для определения подключения. Поле **NAME** в командах вывода всегда определяет идентификатор подключения (**id**).
- **uuid** – уникальная строка идентификации, назначенная системой профилю подключения. **uuid** может быть использован в командах **nmcli connection** для определения подключения.



Инструмент **nmcli** имеет интерактивный редактор подключений. Чтобы использовать его введите следующую команду:

```
# nmcli con edit
```

Далее будет запрошен ввод корректного типа подключения из отраженного списка. после ввода типа подключения запустится строка **nmcli**. Подключение может быть передан команде **nmcli** con edit при запуске, чтобы сразу попасть в строку **nmcli**. Используйте следующий формат для редактирования существующих профилей подключения:

```
nmcli con edit [id | uuid | path] ID
```

Для редактирования нового профиля подключения:

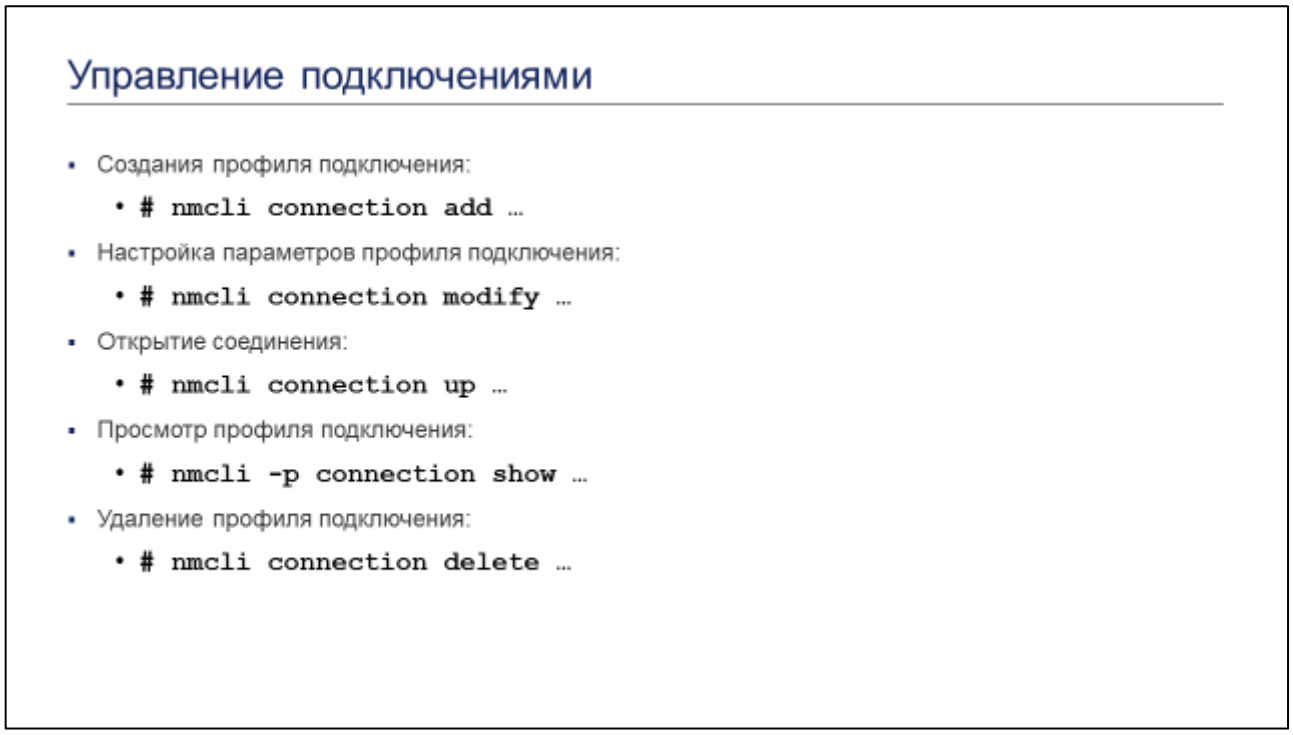
```
nmcli con edit [type НВОЫЙ-ТИП-ПОДКЛЮЧЕНИЯ] [ИМЯ-ПОДКЛЮЧЕНИЯ ИМЯ-НОВОГО-ПОДКЛЮЧЕНИЯ]
```

При вводе **help** в строке **nmcli** отображается список доступных. Команда **describe** выводит описание параметров и их свойств:

```
describe ПАРАМАТР.СВОЙСТВО
```

Пример:

```
nmcli> describe team.config
```



Создание и изменение профиля подключения.

Профиль подключения содержит информацию о свойствах подключения, необходимую для подключения к источнику данных.

Для создания нового профиля для NetworkManager при помощи **nmcli**:

```
nmcli c add {АРГУМЕНТЫ}
```

nmcli с **add** принимает два различных типов параметров:

- Имена свойств (Property Names).

Имена, которые NetworkManager используя для внутреннего описания подключения. Наиболее важные:

  - **connection.type**

```
nmcli c add connection.type bond
```

  - **connection.interface-name**

```
nmcli c add connection.interface-name eth0
```

  - **connection.id**

```
nmcli c add connection.id "RedOS Connection"
```

Больше информации о свойствах и их настройках можно найти на странице руководства **nm-settings(5)**.

- Имена псевдонимов (Aliases Names).

Удобные для чтения имена, которые переводятся во внутренние свойства. Наиболее популярные:

  - **type** (Свойство **connection.type**).

```
# nmcli c add type bond
```

- **ifname** (Свойство **connection.interface-name**).

```
# nmcli c add ifname eth0
```

- **con-name** (Свойство **connection.id**).

```
# nmcli c add con-name "RedOS Connection"
```

В предыдущих версиях **nmcli** для создания подключения требовалось использовать псевдонимы. например, для создания подключения со следующими параметрами **ifname eth0** и **con-name RedOS Connect** требовалось:

```
# nmcli c add type ethernet ifname eth0 con-name "RedOS Connect"
```

В более новых версиях, можно использовать как имена свойств, так и псевдонимы одновременно.

Следующие примеры корректные делают одно и тоже:

```
# nmcli c add type ethernet ifname eth0 con-name " RedOS Connect " ethernet.mtu
1600
# nmcli c add connection.type ethernet ifname eth0 con-name "RedOS Connect "
ethernet.mtu 1600
# nmcli c add connection.type ethernet connection.interface-name eth0
connection.id "RedOS Connect " ethernet.mtu 1600
```

Аргументы отличаются в соответствии с типом подключения. только аргумент **type** обязательные для всех типов подключений и **ifname** обязательные для всех типов подключения за исключением **bond**, **team**, **bridge** и **vlan**.

Тип подключения: type *ИМЯ_ТИПА*. Пример:

```
# nmcli c add type bond
```

Интерфейс для привязки к подключению: **ifname** *ИМЯ_ИНТЕРФЕЙСА*. Пример:

```
# nmcli c add ifname ИМЯ_ИНТЕРФЕЙСА type ethernet
```

Для изменения одного или нескольких свойств профиля подключения используется следующая команда:

```
# nmcli c modify
```

Например, чтобы изменить **connection.id** с **RedOS Connect** на **RedOS Connection** и **connection.interface-name** на **eth1**, необходимо выполнить следующую команду:

```
# nmcli c modify "RedOS Connect" connection.id "RedOS Connection"
connection.interface-name eth1
```

ПРИМЕЧАНИЕ

Рекомендуется использовать имена свойств (Property Names). Псевдонимы (Aliases) используются только в целях обеспечения совместимости.

Дополнительно чтобы установить MTU проводной сети в значение 1600 выполните следующую команду:

```
# nmcli c modify "RedOS Connection" ethernet.mtu 1600
```

Чтобы изменения, сделанные при помощи nmcli, вступили в силу, вновь активируйте подключение при помощи, следующей команды:

```
# nmcli con up ИМЯ-ПОДКЛЮЧЕНИЯ
```

Пример:

```
# nmcli con up RedOS Connection
```

Подключение к сети.

Чтобы вывести список доступных на данный момент подключений, воспользуйтесь следующей командой:

```
# nmcli con show
NAME  UUID                                  TYPE      DEVICE
eth0  ee2cb4a9-6caf-4674-8f51-979d3d6a233f  ethernet  eth0
eth1  a6961083-d4ac-429e-935b-0efd0bfef5cb  ethernet  --
eth2  24e7f6b3-8e7d-4d33-8063-1a7b1bcece06  ethernet  --
eth3  9b63cadc-fb2a-43de-8dda-7e1b8329cfa3  ethernet  --
```

Обратите внимание что поле NAME в выводе всегда обозначает идентификатор подключения (Имя, назначаемое пользователем). Это не имя интерфейса, даже если они выглядят одинаково.

Добавление проводного подключения обозначает создание профиля конфигурации, который затем назначаются устройству. Перед созданием нового профиля необходимо посмотреть список доступных устройств:

```
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     eth0
eth1    ethernet  disconnected  --
eth2    ethernet  disconnected  --
eth3    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

Чтобы сделать устройство неуправляемым (Unmanaged) при помощи NetworkManager:

```
# nmcli device set ИМЯ-ИНТЕРФЕЙСА managed no
```

Пример:

```
# nmcli  device set eth1 managed no
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     eth0
eth2    ethernet  disconnected  --
eth3    ethernet  disconnected  --
eth1    ethernet  unmanaged     --
lo      loopback  unmanaged     --
```

ПРИМЕЧАНИЕ

Когда устройство делается неуправляемым (Unmanaged) NetworkManager более не управляет им. Однако устройство остается подключенным.

Добавление динамического подключения проводной сети.

Чтобы добавить профиль конфигурации сети Ethernet с динамической настройкой IP, позволяющий DHCP назначить сетевую конфигурацию, используется команда следующего вида:

```
# nmcli connection add type ethernet con-name ИМЯ-ПОДКЛЮЧЕНИЯ ИМЯ-УСТРОЙСТВА
```

Пример создания профиля динамического подключения с именем **RedOS**:

```
# nmcli connection add type ethernet con-name RedOS ifname eth1
Connection 'll-100' (bfe7392f-cbff-4492-a1ca-5bd19dd93eef) successfully added.
```

Чтобы открыть соединение по сети Ethernet:

```
# nmcli connection up RedOS
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/3)
```

Проверим статус устройств и подключений:

```
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     eth0
eth1    ethernet  connected     RedOS
eth2    ethernet  disconnected  --
eth3    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

Настройка динамического подключения проводной сети.

Чтобы изменить имя хоста, отправляемого хостом на сервер DHCP, необходимо изменить свойства **dhcp-hostname**:

```
# nmcli connection modify RedOS ipv4.dhcp-hostname srv01 ipv6.dhcp-hostname srv01
```

Чтобы изменить идентификатор клиента (Client ID), отправляемого хостом на сервер DHCP, необходимо изменить свойство **dhcp-client-id**:

```
# nmcli connection modify RedOS ipv4.dhcp-client-id 00155d46d2aa
```

Не существует опции **dhcp-client-id** для IPv6, dhclient создаёт идентификатор для IPv6. подробнее процедура описана странице руководства **dhclient(8)**.

Чтобы игнорировать сервер DNS присылаемые хосту сервером DHCP, необходимо изменить свойство **ignore-auto-dns**:

```
# nmcli connection modify RedOS ipv4.ignore-auto-dns yes ipv6.ignore-auto-dns yes
```

Дополнительная информацию о свойствах и их настройках доступна на странице руководство **nm-settings(5)**.

Пример настройки динамического подключения проводной сети при помощи интерактивного редактора:

```
# nmcli connection edit type ethernet con-name 802-3-eth

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.
```

```
You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-
1x, dcb, ipv4, ipv6, tc, proxy
nmcli> set ipv4.method auto
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an immediate
activation of the connection.
Do you still want to save? (yes/no) [yes] yes
Connection '802-3-eth' (6bf38999-ea37-4bd3-bf69-55978ffd2d9b) successfully saved.
nmcli> quit
```

Действие по умолчанию – это сохранение профиля подключения на постоянной основе. Если необходимо, профиль может храниться только в памяти до следующей перезагрузки, для этого используется команда **save temporary**.

Для удаления подключения воспользуйтесь следующей командой:

```
# nmcli connection delete RedOS
```

Добавление статического включение проводной сети.

Для добавления подключение сети Ethernet со статической конфигурации IPv4:

```
# nmcli connection add type ethernet con-name RedOS ifname eth1 ip4
192.168.0.201/24 gw4 192.168.0.1
Connection 'RedOS' (41ba2e20-f55e-466f-9616-844dbfcb4306) successfully added.
```

Информация об адресе и шлюзе IPv6 может быть добавлена при помощи опций ip6 и gw6.

Чтобы установить два адреса серверов DNS:

```
# nmcli connection modify RedOS ipv4.dns "192.168.0.1 172.20.70.253"
```

Обратите внимание что данная команда заменяет ранее установленные серверы DNS, чтобы вместо этого добавить сервер DNS Используйте префикс **+**:

```
# nmcli connection modify RedOS +ipv4.dns "8.8.8.8"
```

Чтобы открыть соединение по сети Ethernet:

```
# nmcli connection up RedOS
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/3)
```

Проверим статус устройств и подключений:

```
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     eth0
eth1    ethernet  connected     RedOS
eth2    ethernet  disconnected  --
eth3    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

Для просмотра подробной информации, а вновь созданном и настроено подключение выполните следующую команду:

```
# nmcli -p connection show RedOS
===============================================================================
                      Connection profile details (RedOS)
===============================================================================
connection.id:                          RedOS
```

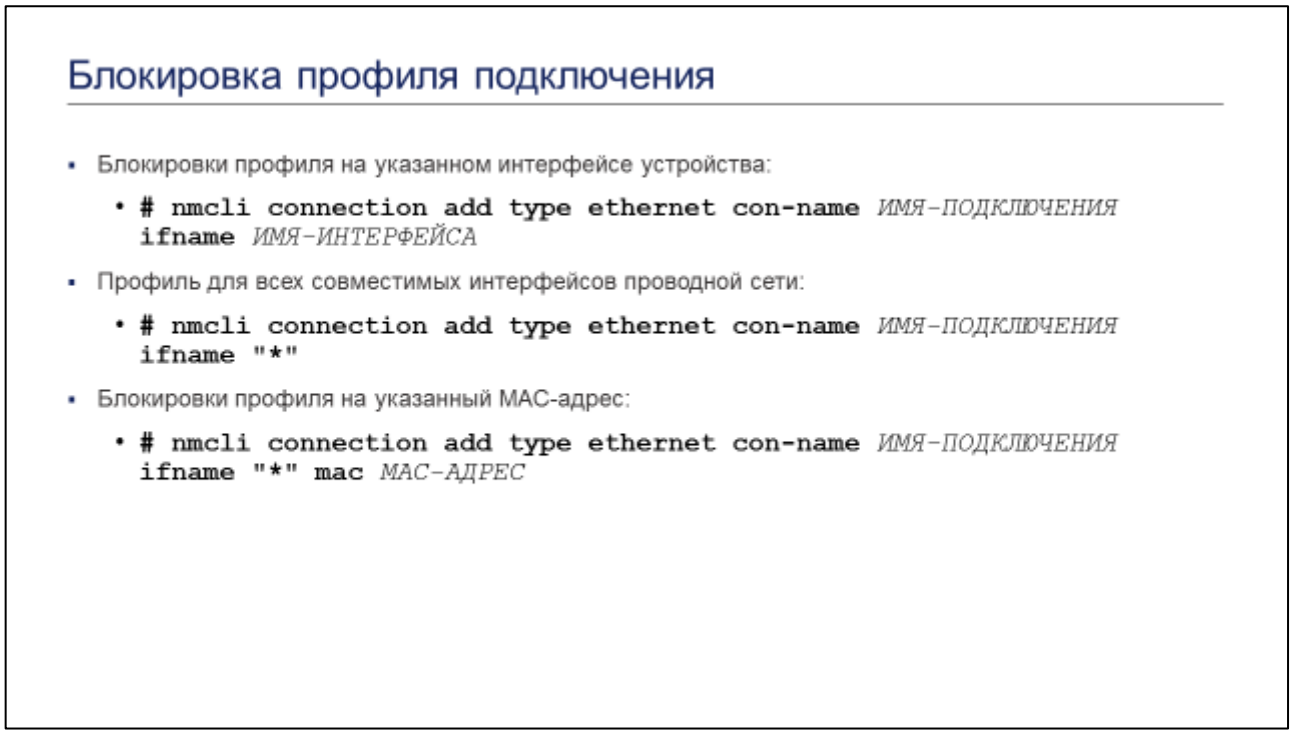
```
connection.uuid:                        41ba2e20-f55e-466f-9616-844dbfcb4306
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              eth1
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.auth-retries:                -1
connection.timestamp:                   1551500717
connection.read-only:                   no
connection.permissions:                 --
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.autoconnect-slaves:          -1 (default)
connection.secondaries:                 --
connection.gateway-ping-timeout:        0
connection.metered:                     unknown
connection.lldp:                        default
lines 1-23
```

Пример настройки статического подключения проводной сети при помощи интерактивного редактора:

```
# nmcli connection edit type ethernet con-name 802-3-ethernet

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-
1x, dcb, ipv4, ipv6, tc, proxy
nmcli> set ipv4.addresses 192.168.0.201/24
nmcli> save temporary
Saving the connection with 'autoconnect=yes'. That might result in an immediate
activation of the connection.
Do you still want to save? (yes/no) [yes] yes
Connection 'demo' (772d025e-fb2a-4f57-87ae-19e374e6ea01) successfully saved.
nmcli> save
Connection 'demo' (772d025e-fb2a-4f57-87ae-19e374e6ea01) successfully updated.
nmcli> quit
```

NetworkManager будет устанавливать его внутренний параметр **connection.autoconnect** в значение yes. Также NetworkManager будет записывать параметры в файл **/etc/sysconfig/network-scripts/ifcfg-RedOS**.

Обратите внимание что изменения внесены вручную в файл **ifcfg** автоматический не уведомляют NetworkManager до тех пор, пока интерфейс не будет перезапущен.



Для блокировки профиля на указанном интерфейсе устройства:

```
# nmcli connection add type ethernet con-name ИМЯ-ПОДКЛЮЧЕНИЯ ifname ИМЯ-ИНТЕРФЕЙСА
```

Чтобы сделать профиль для использования совместимыми интерфейсами проводной сети:

```
# nmcli connection add type ethernet con-name ИМЯ-ПОДКЛЮЧЕНИЯ ifname "*"
```

Обратите внимание что используется аргумент **ifname**, даже когда не планируется установка определенного интерфейса. Символ подстановки ***** используется чтобы указать, что профиль может быть использован любым совместимым устройством.

Чтобы заблокировать профиль на указанный MAC-адрес:

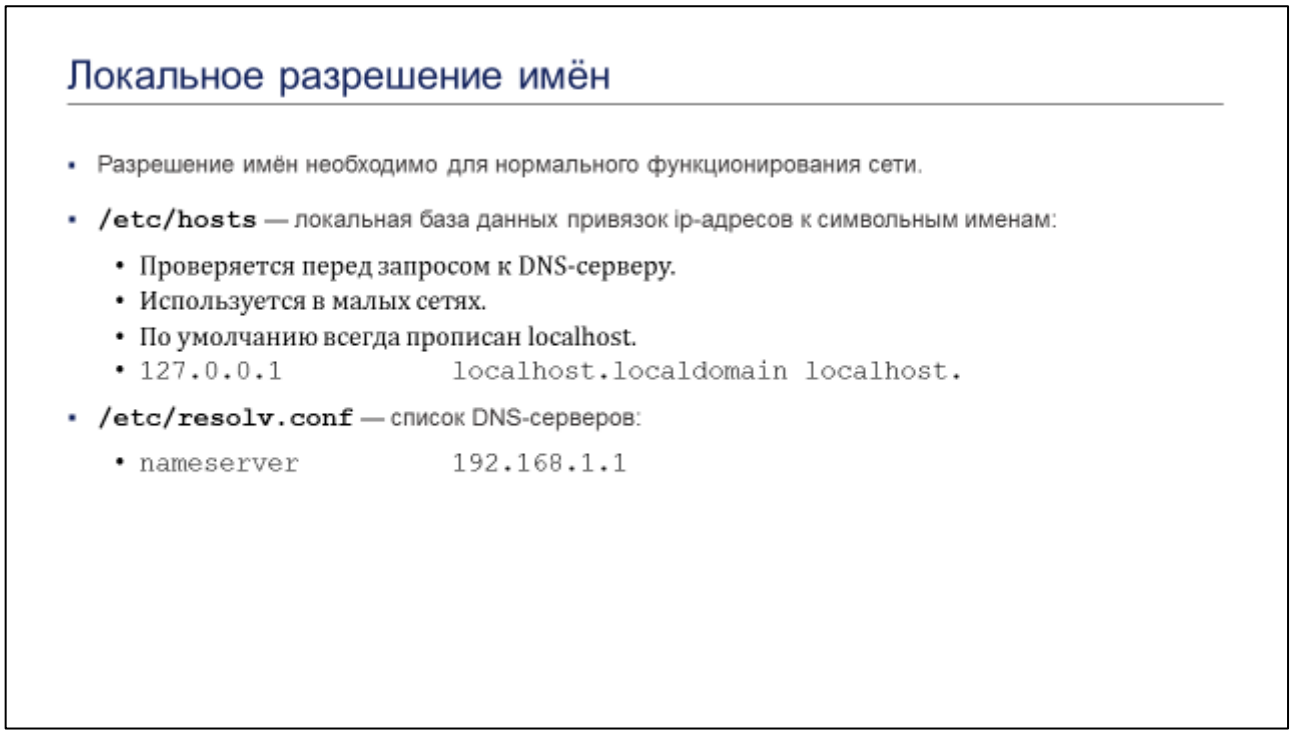
```
# nmcli connection add type ethernet con-name ИМЯ-ПОДКЛЮЧЕНИЯ ifname "*" mac 00:00:5E:00:53:00
```

Установленная документация:

- **ip(8)** – описывает синтаксис утилиты ip.
- **nmcli(1)** – описывает инструмент командной строки NetworkManager.
- **nmcli-examples(5)** – содержит примеры команд **nmcli**.
- **nm-settings(5)** – описывает свойства и настройки NetworkManager.

Онлайн документация:

- RFC 1518: Classless Inter-Domain Routing (CIDR) – описывает назначение адресов и стратегия агрегации CIDR, в том числе подсети переменной длины.
- RFC 1918: Address Allocation for Private Internets – описывает диапазоны адресов IPv4 зарезервированных для частного использования.
- RFC 3330: Special-Use IPv4 Addresses – описывается глобальные и прочие специализированные блоки адресов IPv4 которые были назначены.



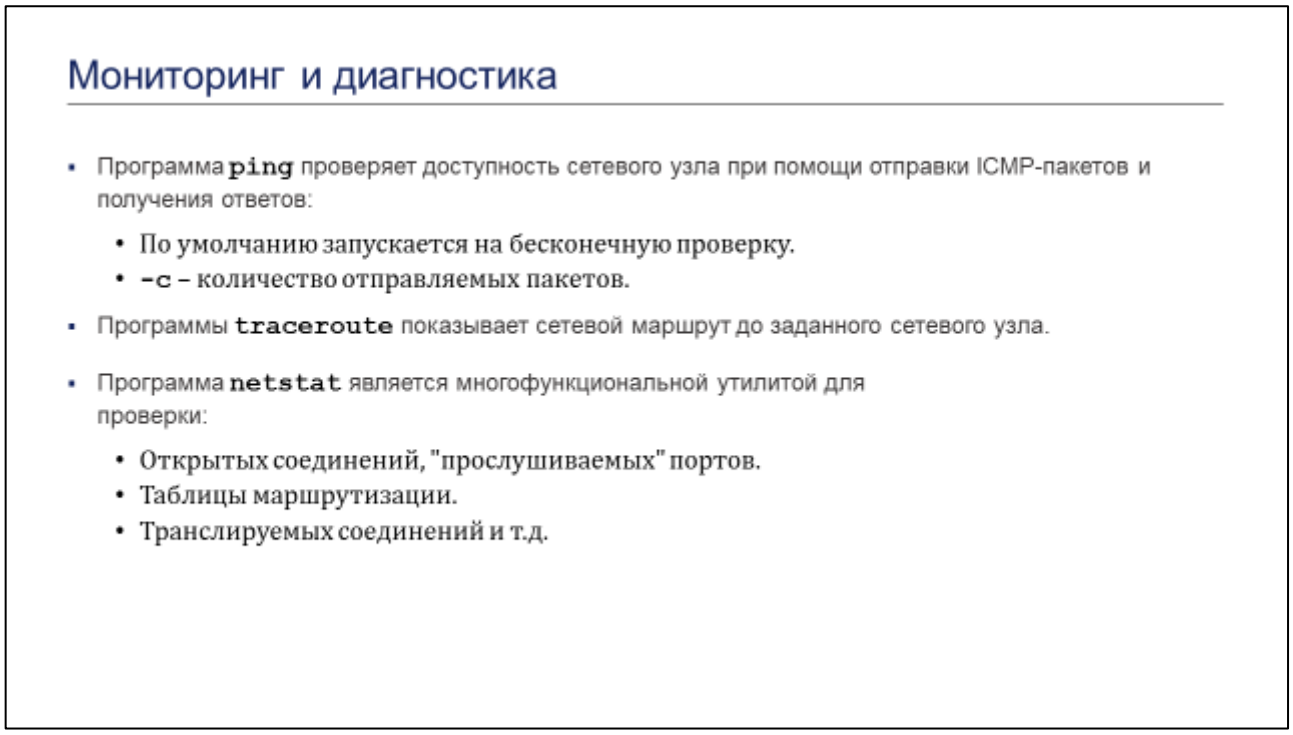
Система разрешения имён сделала работу в сети удобной и массовой. Без разрешения имён не обходится практически ни один запрос.

Как и в операционной системе MS Windows, RedOS имеет файл **/etc/hosts**, в котором указывается соответствие имён ip-адресам. Естественно, что использование такого файла в качестве основной системы разрешения имён вызывает огромные сложности, именно поэтому файл **/etc/hosts** используется только для разрешения имени **localhost**:

```
# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       tuxnote localhost.localdomain   localhost
```

Учитывая, что RedOS является системой, очень сильно ориентированной на сетевую работу, то можно смело предложить, что она умеет работать и с сервером имён, называемым DNS (Domain Name System), чтобы указать системе адрес сервера имён, необходимо внести изменения в файл **/etc/resolv.conf**:

```
# cat /etc/resolv.conf
nameserver 10.10.10.10
nameserver 192.168.1.2
```



Стандартные средства диагностики работоспособности сети используются администраторами очень часто. К стандартным средствам можно отнести не один десяток программ, но мы остановимся на 3-х основных, с помощью которых вы всегда сможете установить факт сбоя в сети, чтобы в дальнейшем прибегнуть к соответствующим способам восстановления работоспособности.

Команда **ping** используется для проверки доступности сетевого узла. Программа формирует специальный icmp-пакет, называемый **ECHO_REQUEST** и отправляет его удалённому узлу, ожидая ответ, называемый **ECHO_RESPONSE**. В случае, если ответ не пришёл, то можно сделать вывод о недоступности сетевого узла, но администратор всегда должен помнить, что неполучение ответа говорит лишь о том, что какой-то из пакетов не дошёл до адресата, а это может быть вызвано правилами брандмауэра в любой точке на пути следования пакета.

По умолчанию, команда ping работает бесконечно (ожидая сигнала к завершению, данного пользователем):

```
# ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=1.81 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=1.03 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=1.04 ms
--- 10.10.10.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.036/1.298/1.813/0.366 ms
```

Программа принимает разнообразные аргументы:

- **-b** — позволяет "пинговать" широковещательные адреса;
- **-c** — сколько пакетов необходимо отправить;
- **-q** — тихий режим, будет выведена только итоговая информация.

Программа **traceroute** демонстрирует администратору маршрут до удалённого узла, это необходимо для проверки таблицы маршрутизации и определения проблем. Используются

различные сетевые протоколы, но принцип действия один — посылается tcp-пакет с установленным значением **TTL=1**. Первый же маршрутизатор, получивший такой пакет, отправит ответ, что время жизни пакета истекло, после получения такого ответа, traceroute формирует новый пакет со значением **TTL=2** и так далее, вплоть до узла назначения.

Программа **netstat** – многофункциональный инструмент для просмотра состояния сетевого стека операционной системы, позволяя отслеживать информацию о сетевых интерфейсах, таблице маршрутизации, статистике интерфейса, информацию о транслируемых адресах и т.д.

Чтобы посмотреть открытые в системе сетевые порты, дайте команду:

```
# netstat -na | head -n 10
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:802                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:139                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:5900                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:20465               0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:445                 0.0.0.0:*                   LISTEN
tcp        0      0 10.10.10.44:42136           66.102.9.99:80
ESTABLISHED
```

Модуль 6. Инструментальные средства системного администрирования РЕД ОС

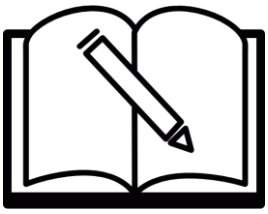
В этом модуле рассматриваются:

Настройка сервера печати CUPS

Настройка и применение OpenSSH.

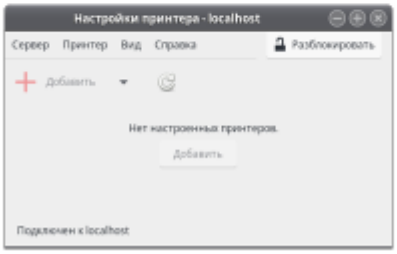
Система ведения журналов Journald

Система ведения журналов Rsyslog



Параметры печати

- Утилита Настройки принтера (Print Settings) предназначена для управления принтерами.
  - **system-config-printer**
  - Базируется на Common Unix Printing Settings (CUPS).
- Сервер печати CUPS предназначен для управления печатью.
  - Использует протокол IPP (Internet Printing Protocol).
  - Основной конфигурационный файл: **/etc/cups/cupsd.conf**.
  - Веб-интерфейс: http://localhost:631/.



Утилита **Print Settings** предназначена для настройки принтера, обслуживания конфигурационных файлов принтеров, буферных каталогов печати и фильтров печати, а также для управления классами принтеров.

Утилита базируется на Common Unix Printing Settings (CUPS).

ПРИМЕЧАНИЕ

Страница руководства **cupsd.conf** описывает настройку сервера CUPS. Она содержит директивы для включения поддержки SSL. Однако, CUPS не позволяет управлять версией используемого протокола. В связи с уязвимостью POODLE SSLv3.0 рекомендуется не использовать встроенные возможности SSL. Надежнее использовать **stunnel** для предоставления безопасного канала и отключить SSLv3.

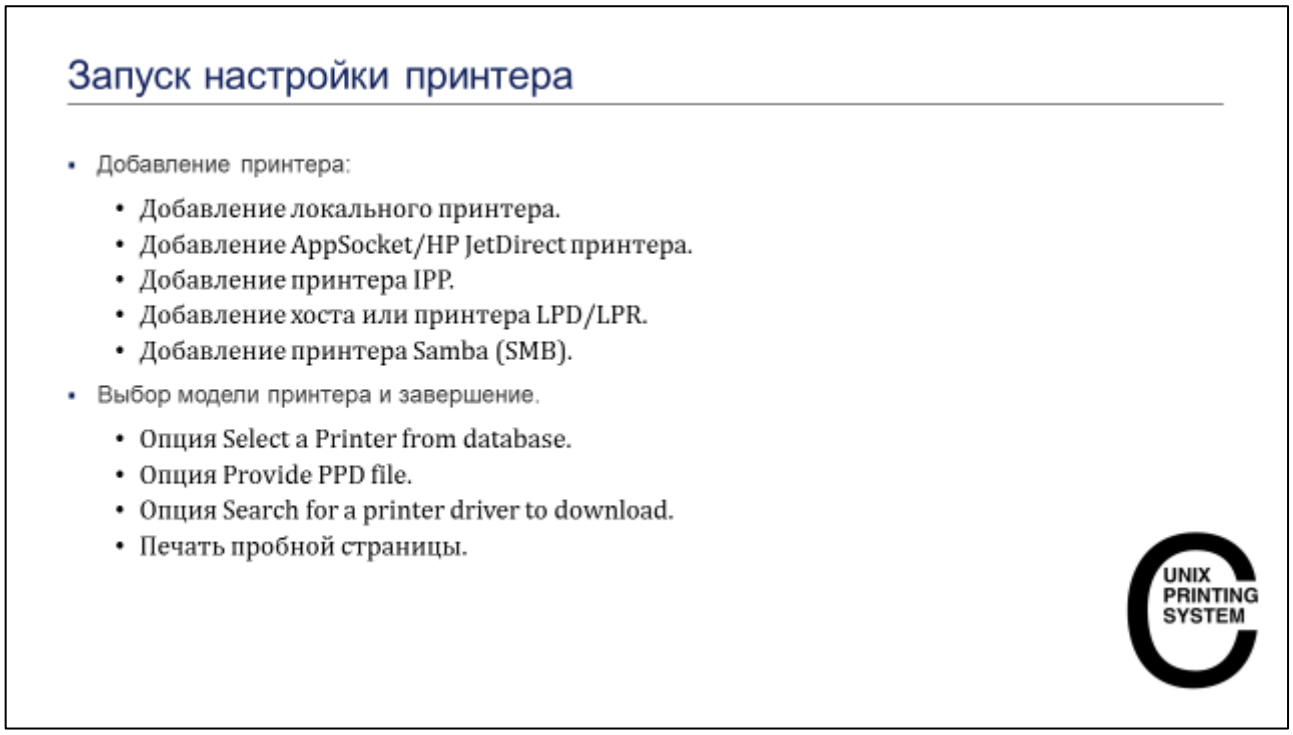
ПРИМЕЧАНИЕ

Можно выполнить аналогичные и дополнительные операции напрямую через веб-приложение CUPS или командную строку. Для доступа к приложению обратитесь к ссылке http://localhost:631/ при помощи веб-браузера. Для получения доступа к руководствам CUPS воспользуйтесь ссылками на вкладке **Home**.

Запуск утилиты настройки параметров печати (Print Settings).

При помощи утилиты настройки Print Settings можно выполнять различные действия на существующих принтерах, устанавливать новые принтеры. Также можно использовать CUPS напрямую (при помощи веб-приложения доступного по ссылке http://localhost:631/).

Для запуска утилиты Print Settings из командной строки, введите **`system-config-printer`** в строку оболочки.

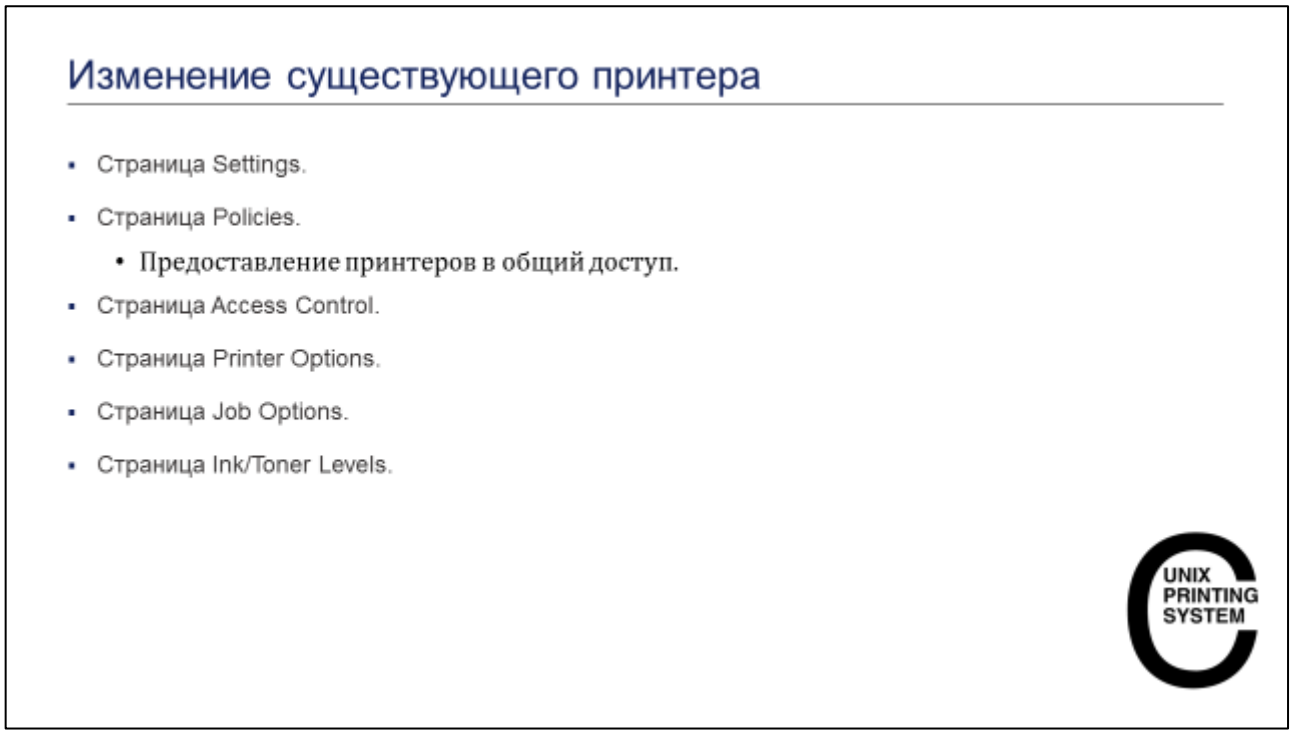


Процесс установки принтера изменяется в зависимости от типа очереди принтера.

Если вы настраиваете локальный принтер, подключенный через USB, принтер обнаруживается и добавляется автоматически. Возможно, будет запрошено подтверждение установки пакета и пароль администратора или root. Локальные принтеры, подключаемые через другие типы портов, и сетевые принтеры нуждаются в ручной настройке.

Процедура запуска ручной настройки принтера:

1. Запустите утилиту Printer Settings.
2. Выберите Server > New > Printer.
3. В диалоговом окне Authenticate введите пароль администратора или пользователя root. Если в первые настраивается удаленный принтер, будет запрошено разрешение на изменение в межсетевом экране (Firewall).
4. Выберите тип подключения принтера и укажите детали в области справа.



Чтобы изменить конфигурацию драйвера принтера, откройте страницу Settings необходимого принтера.

На ней можно изменять параметры принтера, такие как производитель и модель, печатать тестовую страницу, изменять расположение устройства (URL) и так далее.

Страница Policies.

Нажмите кнопку Policies слева чтобы изменить состояние и вывод принтера.

Вы можете выбрать состояния принтера, настроить Error Policy принтера (доступны на выбор прервать задание печати, повторить или остановить при возникновении ошибки).

Также можно создать Banner Page (страницу которая описывает аспекты задания печати, такие как исходный принтер, имя пользователя, создавшего задание и статус печатаемого документа): нажмите Starting Banner или Ending Banner в ниспадающем меню и выберите опции, которые лучше всего описывают природу задания печати (например, конфиденциальность).

Предоставление принтеров в общий доступ.

На странице Policies, вы можете отметить принтер как общий: если принтер общий, пользователи, имеющие доступ к сети, могут его использовать. Чтобы разрешить общий доступ к принтеру, перейдите Server > Settings и выберите Publish shared printers connected to this system.

Убедитесь, что межсетевой экран разрешает входящие соединения TCP на порту 631, порт для протокола Network Printing Server (IPP).

Чтобы разрешить службу IPP в firewalld воспользуйтесь следующей командой:

```
# firewall-cmd --add-service=ipp --permanent
# firewall-cmd --reload
```

В качестве альтернативы, можно воспользоваться графическим инструментом **`firewall-config`**.

Страница Access Control.

На странице Access Control можно изменить уровень доступа пользователя к настроенному принтеру. Выберите один из двух вариантов Allow printing for everyone except these users или Deny printing for everyone except these users, затем определите список пользователей ниже, вводя имена пользователей в поле и нажимая кнопку Add.

Страница Printer Options.

Страница Printer Options содержит различные опции настройки печати, контент этой страницы меняется в зависимости от выбранного принтера. Она содержит общие настройки, параметры печати, бумаги, качество и размеров печати.

Страница Job Options.

Страница Job Options содержит подробности заданий печати. Измените параметры по умолчанию, чтобы применить настраиваемые опции заданий, такие как: число копий, ориентация, количество страниц на листе, масштабирование (увеличение или уменьшение области печати на листе), подробные опции текста и настраиваемые опции заданий.

Страница Ink/Toner Levels.

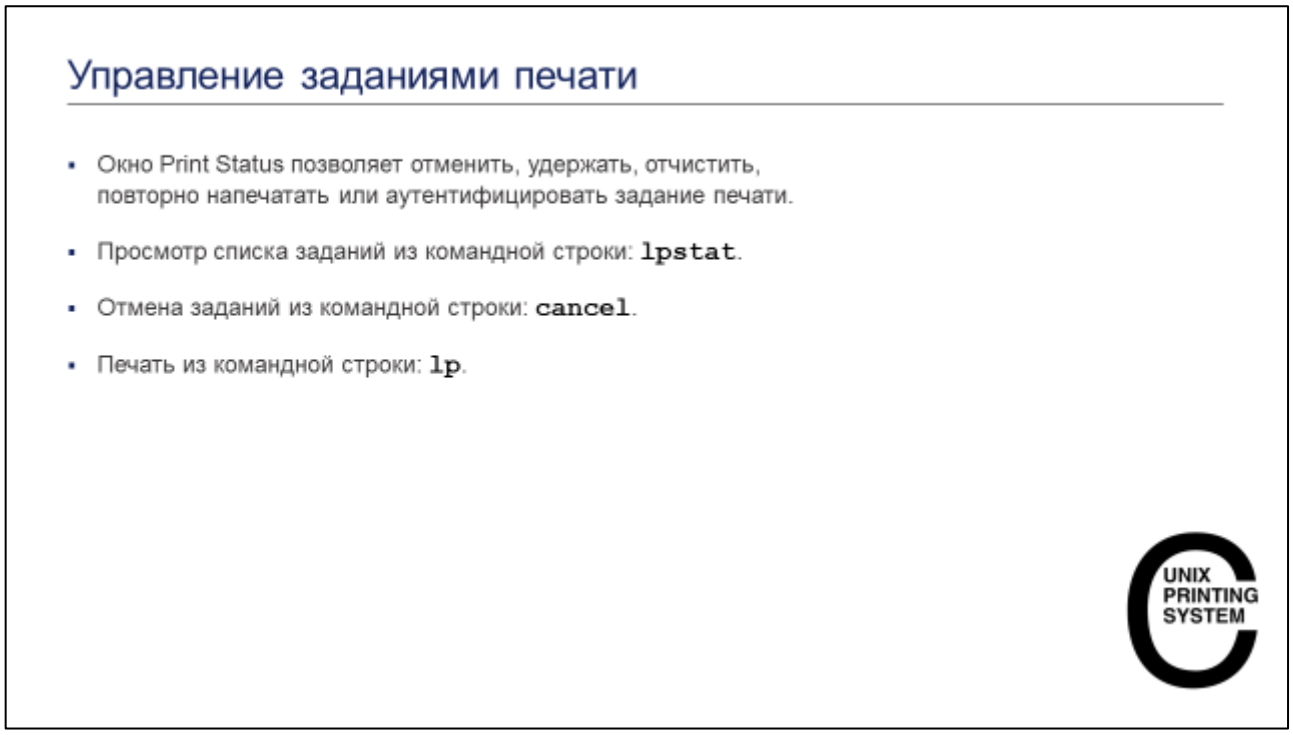
Страница Ink/Toner Levels содержит подробности о статусе тонера, если доступы статусные сообщения принтера.

Удаление принтера.

Для удаления существующего принтера в окне Print Settings, выберите принтера и перейдите Printer > Delete. Подтвердите удаления принтера. Также можно воспользоваться клавишей <Delete>.

Чтобы установить принтер по умолчанию, вызовите меню на принтере и выберите кнопку Set as Default в контекстном меню.

Для изменения параметров принтера, воспользуйтесь следующими страницами:



Когда вы отправляете задание печати на принтер, такое как печать текстового файла из редактора Emacs или изображения из GIMP, задание печати добавляется в очередь буфера печати. Буфер очереди печати – это список заданий печати, которые были отправлены на принтер и информация о каждом запросе печати, такая как статус запроса, номер задания и так далее.

В окружении рабочего стола, в процессе печати, иконка Printer Status отобразится в области уведомлений (Notification Area) на панели. Для проверки статуса задания печати, щелкните иконку Printer Status и отобразится окно Print Status.

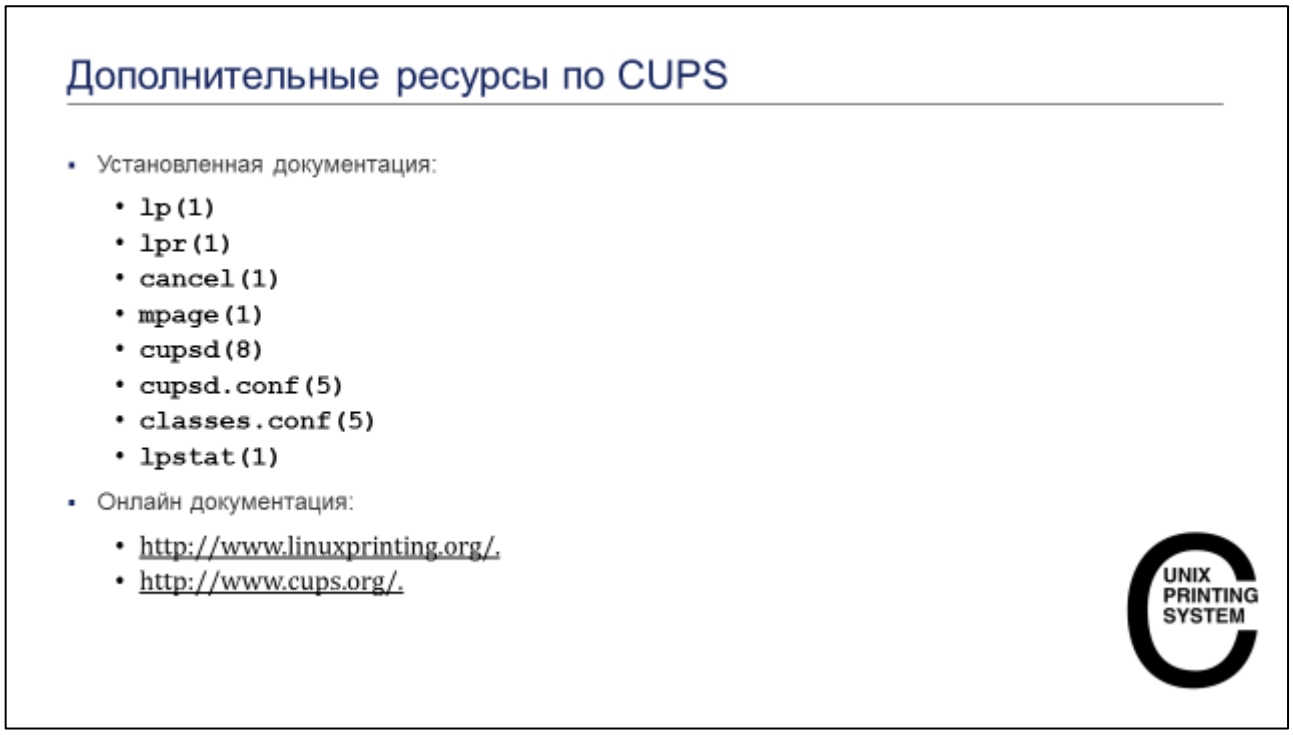
Чтобы отменить, удержать, отчистить, повторно напечатать или аутентифицировать задание печати в Print Status выберите меню Job и щелкните необходимую команду.

Для просмотра списка заданий печати в буфере печати из командной строки, введите команду **lpstat -o**.

```
# lpstat -o
Canon-LBP-1260-1        student            1024   Mon 13 Feb 2017 12:45:42 AM +06
Canon-PIXMA-iP4200-2    student            1024   Mon 13 Feb 2017 12:45:48 AM +06
Canon-PIXMA-iP4200-3    student            1024   Mon 13 Feb 2017 12:46:22 AM +06
Canon-PIXMA-iP4200-4    student           11264   Mon 13 Feb 2017 12:46:44 AM +06
```

Если необходимо отменить задание печати, найдите номер запроса при помощи команды **lpstat** – и затем используйте команду **cancel** *НОМЕР_ЗАДАНИЯ*. Например, **cancel 60** отменит первое задание печати из вывода выше. Невозможно отменить задания печати, запущенные другими пользователями при помощи команды **cancel**. Однако, можно принудительно выполнить удаление любого задания при помощи команды **cancel -U root** *НОМЕР_ЗАДАНИЯ*. Чтобы предотвратить такую отмену, необходимо перевести операционную политику принтера в Authenticated для принудительной аутентификации root.

Также можно печатать файлы напрямую из строки оболочки. Например, команда **lp sample.txt** распечатывает текстовый файл **sample.txt**. Фильтр печать определяет какой тип пришедшего файла и конвертирует в формат понятный принтеру.



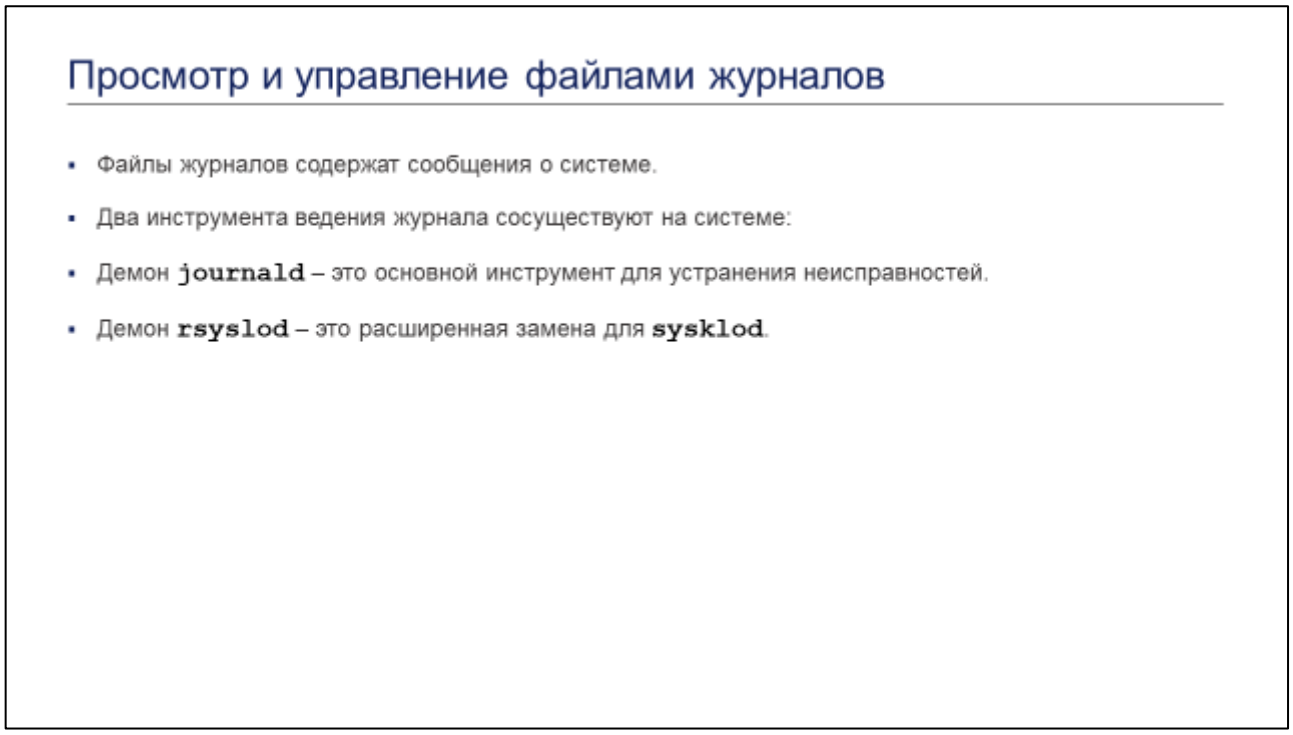
Воспользуйтесь следующими ресурсами для изучения печати в RedOS.

Установленная документация:

- **lp(1)** – страница руководства для команды **lp**, которая позволяет печатать файлы из командной строки.
- **lpr(1)** – страница руководства для команды **lpr**, которая позволяет вам печатать файлы из командной строки.
- **cancel(1)** – страница руководства для утилиты командной строки для удаления заданий из очереди печати.
- **mpage(1)** – страница руководства для утилиты командной строки для печати нескольких страниц на одном листе.
- **cupsd(8)** – страница руководства для демона печати CUPS.
- **cupsd.conf(5)** – страница руководства для конфигурационного файла демона печати CUPS.
- **classes.conf(5)** – страница руководства для конфигурационного файла классов CUPS.
- **lpstat(1)** – страница руководства для команды **lpstat**, которая отображает статусную информацию о классах заданиях и принтерах.

Онлайн документация:

- http://www.linuxprinting.org/ – группа OpenPrinting на веб-сайте Linux Foundation содержит большое количество информации о печати в Linux.
- http://www.cups.org/ – вебсайт CUPS предоставляет документацию, часто задаваемые вопросы и новостные группы о CUPS.



Файлы журналов – это файлы, которые содержат сообщения о системе, в том числе ядре, службах и приложениях к запущенных на ней. Существуют различные файлы журналов для различной информации. Например, файл журнала по умолчанию, файл журнала только для сообщений безопасности и файл журнала для задач планировщика **cron**.

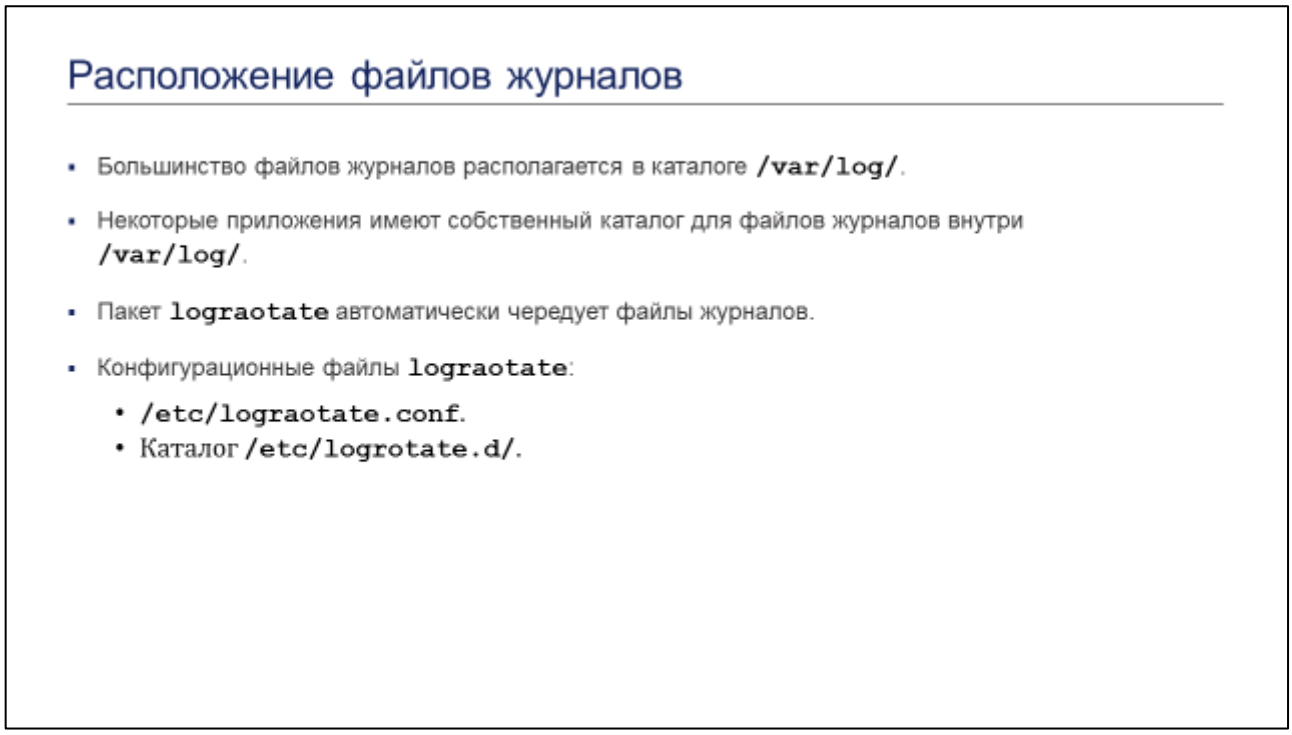
Файлы журналов могут быть очень полезными при устранении проблем с системой, таких как попытка загрузки драйвера ядра или при поиске неавторизованных попыток входа в систему.

Некоторые файлы журналов при помощи демона rsyslod. Демон **rsyslod** – это расширенная замена для **sysklod**, предоставляющая расширенную фильтрацию, защиту передачи сообщений при помощи шифрования, различные опции настройки, модули ввода и вывода, поддержку передачи при помощи протоколов TCP или UDP. Обратите внимание, что **rsyslod** совместим с **syslogkd**.

Также файлами журналов может управлять демон **journald** – компонент **systemd**. Демон **journald** захватывает сообщения **syslog**, сообщения журнала ядра, сообщение начального RAM-диска и начальной загрузки, а также сообщения, выводимые на стандартный вывод и стандартный вывод ошибок, от всех сервисов, индексирует их и делает доступными для пользователя. Собственный формат файла журнала, который является структурированным и индуцированным бинарным файлом, улучшает поиск и быстродействие операций, а также хранит метаданные, такие как штамп времени и идентификаторы пользователей. Файлы журналов, создаваемые **journald**, по умолчанию не постоянны, файлы журналов хранятся только в памяти или маленьком циклическом буфере в каталоге **/run/log/journal**. Объем данных журнала зависит от свободного места в памяти, при достижении лимита объема наиболее старые записи удаляются.

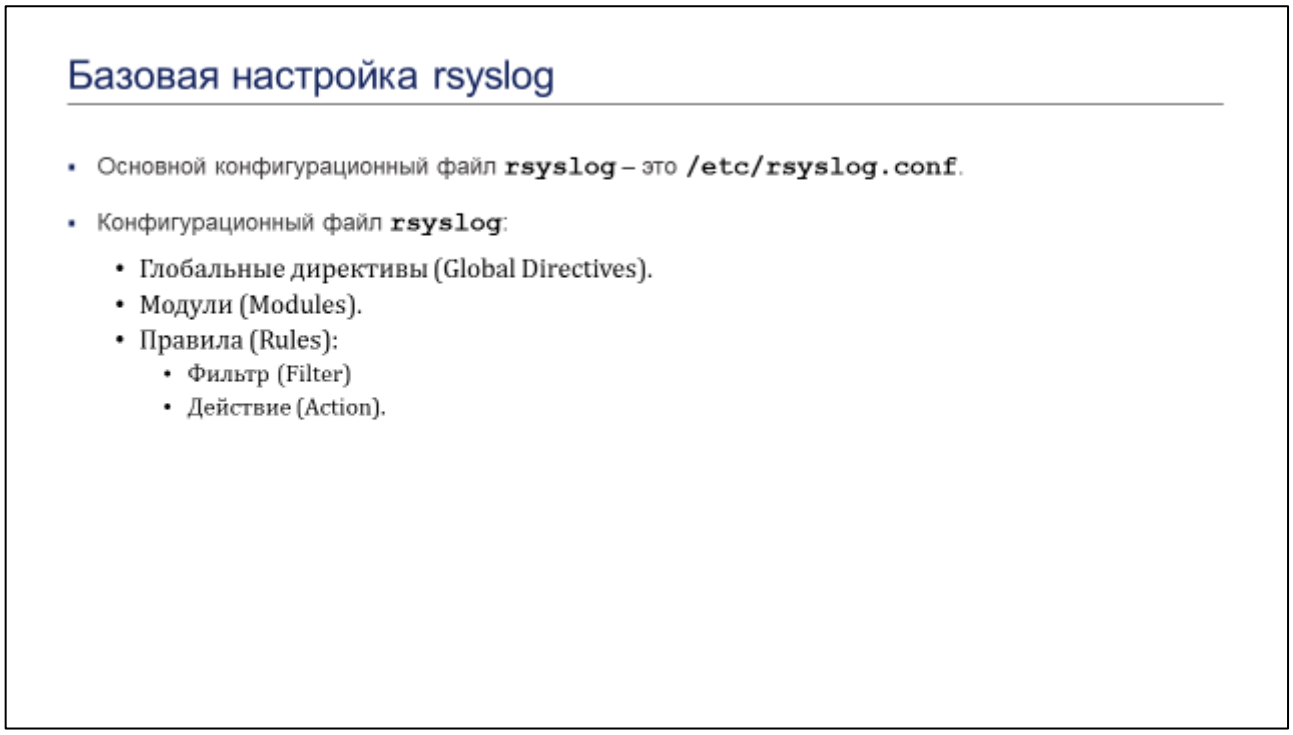
По умолчанию два инструмента ведения журнала сосуществуют на системе. Демон **journald** – это основной инструмент для устранения неисправностей. Он также предоставляет данные, необходимые для создания сообщений журнала. Полученные при помощи **journald** данные перенаправляются в сокет **/run/systemd/journal/syslog**, который может быть использован rsyslod для дальнейшей обработки данных. Однако, **rsyslod** по умолчанию, имеет настоящую

интеграцию через модуль ввода **imjournal**, таким образом избегая использование вышеупомянутого сокета. Можно передавать данные в обратном направлении из **rsyslod** в **journald** при помощи модуля **omjournal**. Интеграция позволяет поддерживать текстовые файлы журналов в целостном формате и убедиться в совместимости с возможными приложениями и конфигурациями, зависящими от **rsyslod**.

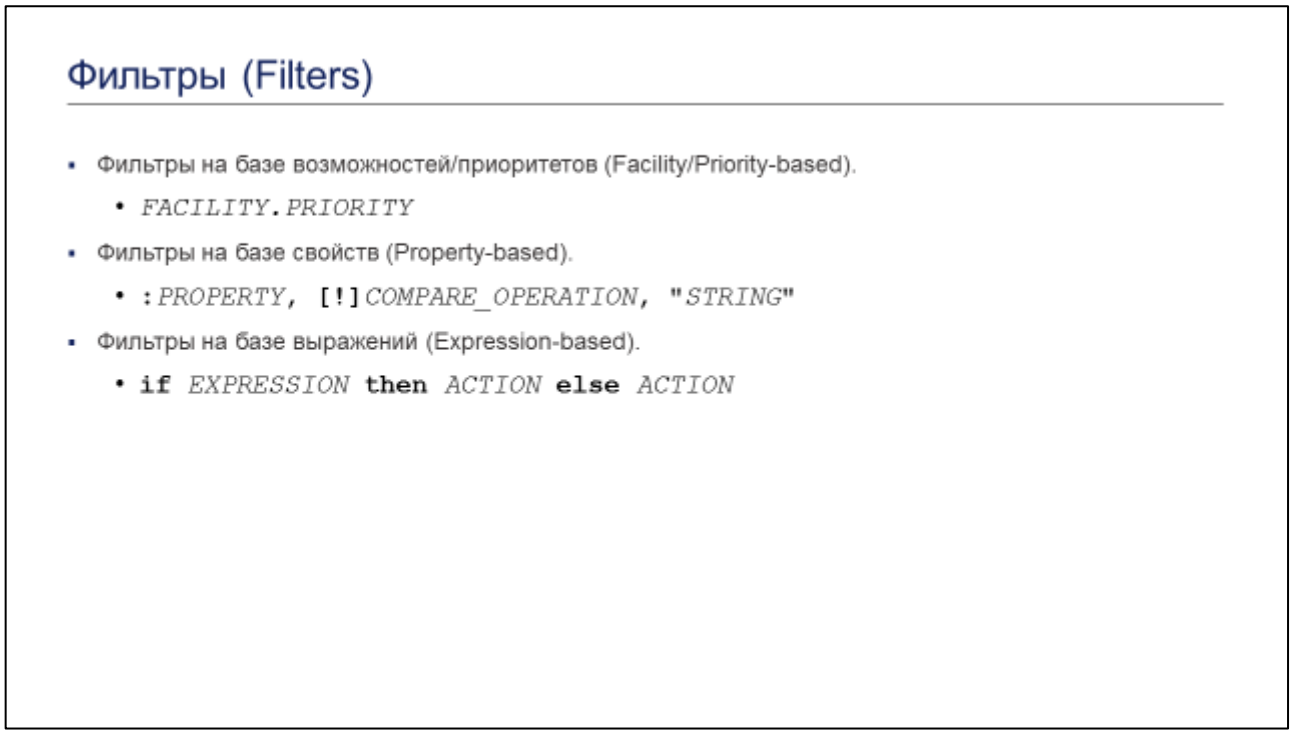


Список файлов журналов, поддерживаемых **rsyslogd** можно найти конфигурационном файле **/etc/rsyslogd.conf**. Большинство файлов журналов располагается в каталоге **/var/log/**. Некоторые приложения, такие как **httpd** и **samba** имеют собственный каталог для файлов журналов внутри **/var/log/**.

В каталоге **/var/log/** многие файлы имеют числа после имен (например, **cron-20180911**). Эти числа представляют собой штамп времени, который добавляется к чередующимся файлам журналов. Файлы журналов чередуются для того чтобы размер файлов не был слишком большим. Пакет **lograotate** содержит задачу **cron**, которая автоматически чередует файлы журналов в соответствии с конфигурационным файлом **/etc/lograotate.conf** и конфигурационными файлами в каталоге **/etc/logrotate.d/**.



Основной конфигурационный файл **rsyslog** – это **/etc/rsyslog.conf**. В нём указываются глобальные директивы (Global Directives), модули (Modules) и правила (Rules), которые состоят из фильтра (Filter) и действия (Action). Дополнительно в конфигурационный файл можно добавить комментарий после знака решетка (**#**).



Правило (Rule) указывается при помощи двух частей. Первая часть – это фильтр (Filter) выбирает подмножество сообщений **syslog**, вторая часть – действие (Action), показывает, что делать с выбранными сообщениями. Для определения правила в конфигурационном файле **/etc/rsyslog.conf** необходимо определить обе части на одной строке и разделить их при помощи пробела или табуляции.

rsyslog предлагает различные пути для фильтрации сообщений syslog в соответствии с выбранными свойствами. Доступные методы фильтрации можно разделить на фильтрацию на базе возможностей/приоритетов (Facility/Priority-based), фильтрацию на базе свойств (Property-based) и фильтрацию на базе выражений (Expression-based).

Фильтры на базе возможностей/приоритетов (Facility/Priority-based).

Наиболее часто используемый и известный путь для фильтрации сообщений syslog – это использование фильтра на базе возможностей/приоритетов, который фильтрует сообщения syslog на базе двух условий: возможности (Facility) и приоритета (Priority), разделенных точкой. Для создания выборщиков используйте следующий синтаксис:

```
FACILITY.PRIORITY
```

Где:

FACILITY указывает подсистему, которая генерирует указанные сообщения syslog. Например, подсистема mail обрабатывает все сообщения syslog, связанные с почтой. FACILITY может быть представлена одним из следующих ключевых слов (или числовым кодом): **kern**(**0**), **user**(**1**), **mail**(**2**), **daemon**(**3**), **auth**(**4**), **syslog**(**5**), **lpr**(**6**), **news**(**7**), **uucp**(**8**), **cron**(**9**), **authpriv**(**10**), **ftp**(**11**) и от **local0** до **local7** (**16** - **23**).

PRIORITY указывает приоритет сообщения **syslog**. Приоритет может быть указан одним из следующих ключевых слов (или при помощи числа): **debug**(**7**), **info**(**6**), **notice**(**5**), **warning**(**4**), **err**(**3**), **crit**(**2**), **alert**(**1**) и **emerg**(**0**).

Вышеуказанный синтаксис выбирает сообщения **syslog** с указанным или более высоким приоритетом. Предшествующее знаку равно, любое ключевое слово приоритета указывает сообщения **syslog** с указанным приоритетом для выборки. Все остальные приоритеты будут проигнорированы. И наоборот, приоритет с предшествующим восклицательным знаком (**!**) выбирает все сообщения **syslog**, кроме сообщения с указанным приоритетом.

В дополнение к ключевым словам, указанным выше, можно использовать звездочку (*****) для определения возможностей или приоритетов (в зависимости от расположения звёздочки до или после точки). Ключевое слово приоритета **none** обозначает возможности без указания приоритета. Определения возможностей и приоритета чувствительны к регистру.

Для определения множества возможностей и приоритетов, необходимо разделить их при помощи запятой (**,**) для определения множества выборщиков на одной строке необходимо разделять их при помощи точки с запятой (**;**). Обратите внимание, что каждый выборщик в поле выборщика может переписать предшествующий, который мог исключить некоторые приоритеты из шаблона.

Примеры фильтров на базе возможностей/приоритетов.

Для выбора всех сообщений **syslog** от ядра с любым приоритетом добавьте следующий текст в конфигурационный файл:

```
kern.*
```

Для выбора всех почтовых сообщений **syslog** с приоритетом **crit** и выше используйте следующую форму записи:

```
mail.crit
```

Для выбора всех сообщений **cron** за исключением приоритетов **info** и **debug**, запишите конфигурацию в следующей форме:

```
cron.!info,!debug
```

Фильтры на базе свойств (Property-based).

Фильтры на базе свойств позволяют фильтровать сообщения **syslog** по любому свойству, такому как время генерации (**timegenerated**) или тегу **syslog** (**syslogtag**). Можно сравнивать каждое из указанных свойств с необходимым значением при помощи одной из операций сравнения. Имена свойств и операции сравнения чувствительны к регистру.

Фильтры на базе свойств должны начинаться с двоеточия (**:**). Для определения фильтра используется следующий синтаксис:

```
:PROPERTY, [!]COMPARE_OPERATION, "STRING"
```

где:

- Атрибут *PROPERTY* указывает необходимое свойство.

- Опциональный восклицательный знак (**!**) отрицает вывод операции сравнения. Прочие Булевы операторы на данный момент не поддерживаются в фильтрах на базе свойств.
- Атрибут *COMPARE_OPERATION* указывает одну или несколько операции сравнения, перечисленных в таблице ниже.
- Атрибут *STRING* указывает значение для сравнения с текстом предоставленным свойством. Данное значение может быть заключено в кавычки ("). Для исключения символов внутри строки, таких как кавычки (") используется символ обратного слеша (\).

Операции сравнения на базе свойств:

Операции сравнения	Описание
contains	Проверяет соответствие предоставленной строки любой части строки предоставленной свойством. Для выполнения не чувствительного к регистру сравнения используется **contains_i**.
isequal	Сравнивает предоставленную строку со всем текстом, предоставленным свойством. Данные два значения должны быть полностью одинаковыми.
startswith	Проверяет соответствие предоставленной строки с началом текста свойства. Для выполнения не чувствительного к регистру сравнения используется **startswith_i**.
regex	Сравнивает предоставленное базовое регулярное выражение POSIX (POSIX BRE) с текстом предоставленным свойством.
ereregex	Сравнивает предоставленное расширенное регулярное выражения POSIX (POSIX ERE) с текстом предоставленным свойством.
isempty	Проверяет, является ли свойство пустым (Empty). Это особенно полезно для нормализованных данных, где некоторые поля могут содержать результаты нормализации.

Примеры фильтров на базе свойств.

Для выбора сообщений **syslog**, которые содержат **Error** в тексте сообщения используется следующая директива:

```
:msg, contains, "error"
```

Следующий фильтр выбирает сообщения **syslog**, полученные от хаста с именем **server02**:

```
:hostname, isequal, "server02"
```

Для выбора сообщений **syslog**, которые не содержат никаких упоминаний о словах **Fatal** и **Error** с любым текстом или без текста между ними используется следующие директива:

```
:msg, !regex, "fatal .* error"
```

Фильтры на базе выражений (Expression-based).

Фильтры на базе выражений выбирают сообщения **syslog**, соответствующие определённой арифметической, сравнительной или строчной операции. Фильтры на базе выражений используют собственный язык написания скриптов **rsyslog**, называемый RainerScript, для построения комплексных фильтров.

Базовый синтаксис фильтра на базе выражения выглядит следующим образом:

```
if EXPRESSION then ACTION else ACTION
```

Где:

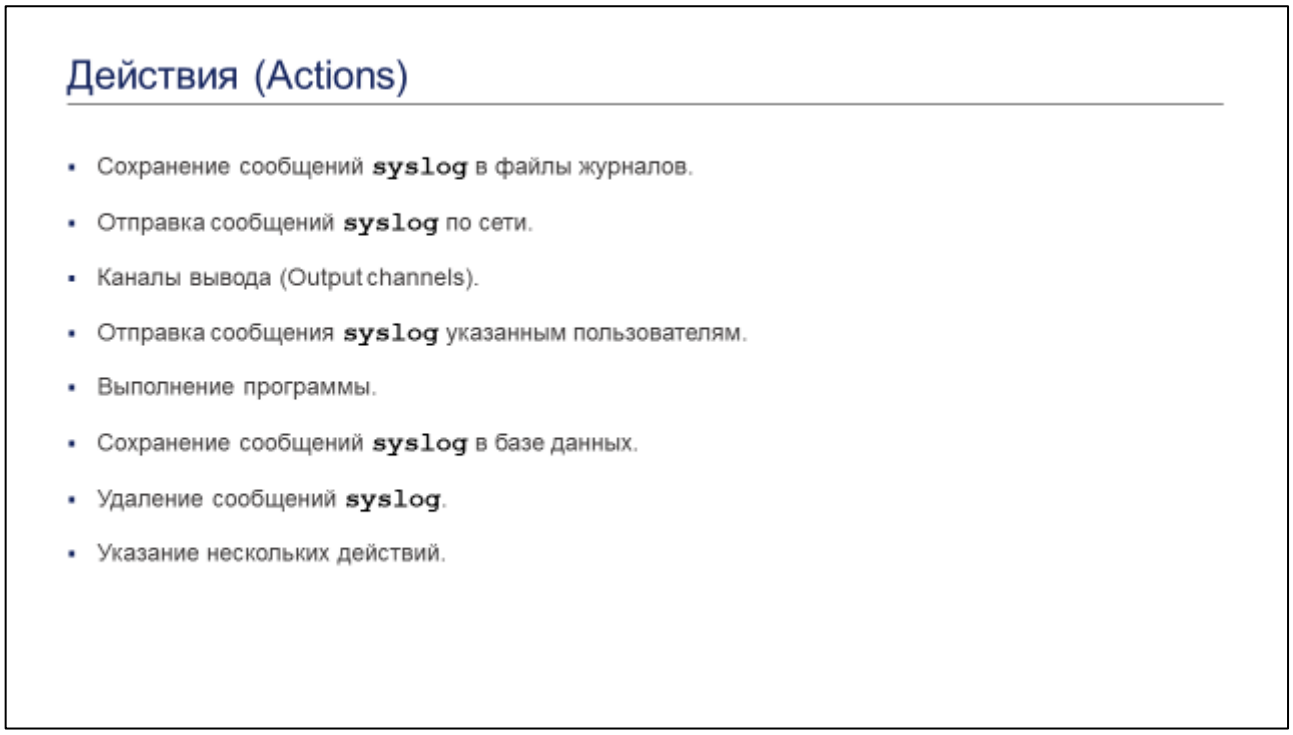
- Атрибут *EXPRESSION* представляет выражение, которое будет оценено, например: **$msg startswith 'DEVNAME'** or **$syslogfacility-text == 'local0'**. Можно указать более одного выражения в одном фильтре при помощи операторов and и or.
- Атрибут **ACTION** представляет действие, которое должно быть выполнено, если выражение возвращает значение **true**. Это может быть отдельное действие или произвольный комплексный скрипт, заключённый в фигурные скобки.
- Фильтры на базе выражений определяются при помощи ключевого слова **if** в начале новой строки. Ключевое слово **then** отделяет выражение от действия. Опционально можно применить ключевое слово **else** для указания действия на случай, когда условие не выполняется.

При помощи фильтров на базе выражений можно вложить множество условий при помощи скрипта, заключённого в фигурные скобки. Скрипт позволяет использовать фильтры на базе возможностей/приоритетов внутри выражения. С другой стороны, использовать фильтры на базе свойства не рекомендуется. RainerScript поддерживает регулярные выражения со специализированными функциями **re_match()** и **re_extract()**.

Примеры фильтров на базе выражений.

Следующее выражение содержит два вложенных условия. Файлы журналов, созданные при помощи программы, называемой **prog1**, разделяются на основе нахождения строки «**test**» в сообщении.

```
if $programname == 'prog1' then {
action(type="omfile" file="/var/log/prog1.log")
if $msg contains 'test' then
action(type="omfile" file="/var/log/prog1test.log")
else
action(type="omfile" file="/var/log/prog1notest.log")
}
```



Действия указывают, что необходимо сделать с сообщением, отобранным при помощи определённого выборщика.

Сохранение сообщений syslog в файлы журналов.

В большинстве случаев действие указывает в какой файл журнала будут сохранены сообщения syslog. Это выполняется при помощи указания пути к файлу после определенного выборщика.

```
FILTER PATH
```

Где *FILTER* используется для указания выборщика, а *PATH* – это путь к целевому файлу.

Например, следующее правило состоит из выборщика, который выбирает все сообщения **syslog** от **cron** и действия, которое сохраняет их в файл журнала **/var/log/cron.log**.

```
cron.* /var/log/cron.log
```

По умолчанию файл журнала синхронизируется каждый раз при генерации сообщения **syslog**. При помощи символа дефиса (**-**) перед именем файла указывается пропуск синхронизации.

```
FILTER -PATH
```

Обратите внимание, если система будет некорректно выключена сразу после попытки записи, это может привести к потере информации. Однако данный параметр может улучшить производительность, особенно если вы запускаете программы, которые генерируют очень подробные сообщения журнала.

Указанный путь может быть статическим (Static) или динамическим (Dynamic). Статические пути к файлам предоставляют фиксированный путь к файлу, как показано в примере выше. Динамические пути к файлам могут отличаться в зависимости от полученного сообщения. Динамические пути предоставляются при помощи шаблона и знака вопроса (**?**):

```
FILTER ?DYNAMIC-FILE
```

Где *DYNAMIC-FILE* – это имя заранее определенного шаблона, который модифицирует пути вывода. Можно использовать префикс дефис (**-**) для отключения синхронизации. Также можно использовать множество шаблонов, разделенных точкой с запятой (**;**).

Если указанный файл – это существующий терминал (Terminal) или устройство **/dev/console**, сообщения syslog отправляются на стандартный вывод (при помощи специального обработчика терминала) или на консоль (при помощи специального обработчика для **/dev/console**), когда используется X Windows System, соответственно.

Отправка сообщений `syslog` по сети.

rsyslog позволяет отправлять и получать сообщения syslog по сети. Даная возможность позволяет администрировать сообщения от множества хостов на одной машине. Для пересылки сообщений syslog на удалённую машину воспользуйтесь следующим синтаксисом:

```
@[(z<NUMBER>)]<HOST>:[<PORT>]
```

Где:

- Знак собака (**@**) определяет, что сообщение syslog пересылаются на хост при помощи протокола UDP. Чтобы использовать протокол TCP, необходимо использовать два знака собака (**@@**).
- Опциональный параметр **z***NUMBER* включает сжатие **zlib** для сообщения **syslog**. Атрибут *NUMBER* указывает уровень компрессии (от минимального **0** до максимального **9**). Целесообразность сжатия автоматически проверяется rsyslogd, если сообщение после сжатия не становится меньше, все сообщения, чей размер меньше 60 байт, не сжимаются.
- Атрибут *HOST* указывает хост, который будет получать выбранные сообщения **syslog**.
- Атрибут *PORT* указывает порт на холсте.

При указании IPv6 адреса в качестве холста, адрес заключается в квадратные скобки (**[]**).

Примеры отправки сообщения syslog по сети.

Для отправки сообщений на хост с IP-адресом **192.168.0.1** по протоколу UDP, используйте:

```
*.* @192.168.0.1
```

Для отправки сообщений на хост example.com при помощи порта **6514** по протоколу TCP, используйте:

```
.* @@example.com:6514
```

Для отправки из сжатых сообщений при помощи **zlib** (с уровнем компрессии **9**) и пересылки их на хост **2001: db8::1** при помощи протокола UDP, используйте:

```
*.* @(z9)[2001:db8::1]
```

Каналы вывода (Output channels).

В основном, каналы вывода используются для указания максимального размера файла журналов, до которого можно расти. Это очень удобно для ротации файлов журналов. По больше части канал

вывода – это коллекция информации о выходных действиях. Каналы вывода определяются при помощи директивы **$outchannel**. Для определения канала вывода в /etc/rsyslog.conf воспользуйтесь следующим синтаксисом:

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```

где:

- Атрибут *NAME* указывает имя канала вывода.
- Атрибут *FILE_NAME* указывает имя файла вывода. Каналы вывода могут записывать только в файлы и не могут записывать в терминалы и другие виды вывода.
- Атрибут *MAX_SIZE* представляет максимальный размер указанного файла (*FILE_NAME*), до которого можно расти. Данное значение указывается в байтах.
- Атрибут *ACTION* указывает действие, которое выполняется при достижении максимального размера, определённого в *MAX_SIZE*.

Для использования определенного канала вывода в качестве действия внутри правила используется следующий синтаксис:

```
FILTER :omfile:$NAME
```

Примеры ротации журналов при помощи канала вывода.

В первую очередь канал вывода определяется при помощи директивы $outchannel:

```
$outchannel log_rotation, /var/log/test_log.log, 104857600,
/home/joe/log_rotation_script
```

Зачем необходимо использовать его в правиле:

```
.* :omfile:$log_rotation
```

При достижении лимита (в примере это 100 MB), выполняется скрипт **/home/joe/log_rotation_script**. Данный скрипт может содержать всё что угодно: перемещение файла в другой каталог, редактирование определённого контента или простого ударение.

Отправка сообщения syslog указанным пользователям.

rsyslog может отправлять сообщения **syslog** определённым пользователям, которым необходимо отправить сообщение, при помощи указания имени пользователя. Для указания нескольких пользователей используется разделитель – запятая (**,**). Для отправки сообщения всем пользователям, которые вошли в систему используется звездочка (*****).

Выполнение программы.

rsyslog позволяет выполнить программу для выбранных сообщений **syslog** и использовать вызов **system()** для выполнения программы в оболочке. Для указания программы, которая должна выполняться, используется префикс в виде символа крышки (**^**), далее необходимо указать шаблон, который форматирует полученное сообщение и передаёт его указанному исполняемому файлу в качестве параметра командной строки.

```
FILTER ^EXECUTABLE; TEMPLATE
```

Вывод условия *FILTER* обрабатывается программой *EXECUTABLE*. Данная программа может быть любым исполняемым файлом. *TEMPLATE*, заменяется именем шаблона форматирования.

Пример выполнения программы.

В примере ниже выбирается любое сообщение **syslog** с любым приоритетом, форматируется при помощи шаблона **template** и передается в качестве параметра в программу **test-program**, которая затем выполняется с предоставленным параметром:

```
*.* ^test-program;template
```

ПРИМЕЧАНИЕ

Когда сообщение принимается от любого хоста, и используются действия выполнения в оболочке – система подвергается уязвимости инъекции команды. Атакующий может попробовать включить исполняемые команды в программу, указанную для выполнения в действии. Чтобы избежать подобных угроз безопасности настоятельно рекомендуется детально рассмотреть необходимость использования действия выполнения в оболочке.

Сохранение сообщений syslog в базе данных.

Выбранные сообщения syslog могут быть напрямую записаны в таблицу базы данных при помощи действия средства записи базы данных (Database Writer). Средство записи базы данных используют следующий синтаксис:

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;[TEMPLATE]
```

Где:

- *PLUGIN* вызывает указанную надстройку, которая обрабатывает запись в базу данных (например, настройка **ommysql**).
- Атрибут *DB_HOST* указывает имя хоста базы данных.
- Атрибут *DB_NAME* указывает имя базы данных.
- Атрибут *DB_USER* указывает имя пользователя базы данных.
- Атрибут *DB_PASSWORD* указывает пароль для пользователя базы данных.
- Атрибут *TEMPLATE* указывает опционально используемый шаблон, который изменяет сообщения **syslog**.

ПРИМЕЧАНИЕ

На данный момент **rsyslog** предоставляет поддержку только для MySQL (MariaDB) и PostgreSQL. Для использования средств записи баз данных MySQL и PostgreSQL необходимо установить пакеты **rsyslog-mysql** и **rsyslog-pgsql** соответственно. Дополнительно, необходимо указать загрузку соответствующих модулей в конфигурационном файле **/etc/rsyslog.conf**:

```
module(load="ommysql") # Output module for MySQL support
module(load="ompgsql") # Output module for PostgreSQL
support
```

В качестве альтернативы можно использовать общий интерфейс баз данных предоставляемый модулем **omlibdb**. Данный модуль поддерживает Firebird, Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL.

Удаление сообщений syslog.

Для удаления выбранных сообщений используется **stop**.

Действие удаления как правило используется для фильтрации сообщений перед выполнением последующей обработки. Это может быть эффективным если необходимо пропустить некоторые повторяющиеся сообщения, которые в противном случае заполнят файлы журналов. Результат действия удаления зависит от того, где оно указано в конфигурационном файле, для наилучшего результата необходимо разместить действие в начале списка действий. Обратите внимание, что если сообщение удалено, то оно не может быть получено в последующих строках конфигурационного файла.

Следующее правило удаляет все сообщения, которые соответствуют фильтру **local5.***:

```
local5.* stop
```

В следующем примере удаляются все сообщения syslog, полученные от cron:

```
cron.* stop
```

ПРИМЕЧАНИЕ

В версиях, предшествующих **rsyslog 7** символ тильда (**~**) использовалась вместо **stop** для удаления сообщений **syslog**.

Указание нескольких действий.

Для каждого выборщика, можно указать несколько действий. Для указания нескольких действий они записываются на отдельных строках, в начале строки указывается символ амперсанд (&):

```
FILTER ACTION
& ACTION
```

```
& ACTION
```

Указание нескольких действий улучшает общую производительность, так как исключает повторную проверку одного и того же фильтра выборщика.

Пример указания нескольких действий.

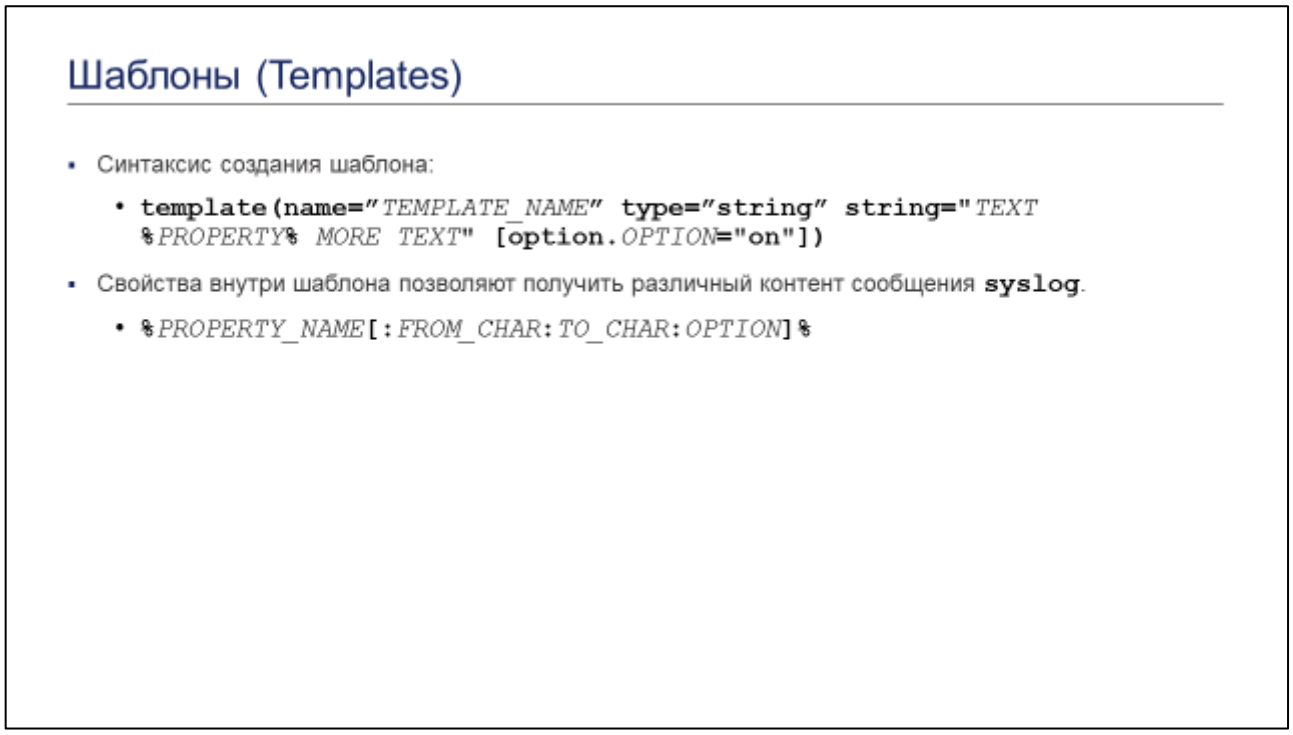
В следующем примере сообщения **syslog** от ядра с критическим (**crit**) приоритетом отправляются пользователю **user1**, обрабатываются шаблоном temp и передаются в исполняемый файл **test-program**, а также пересылаются на хост **192.168.0.1** по протоколу UDP.

```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

Любое действие может сопровождаться шаблоном, который форматирует сообщение. Для указания шаблона, после действия устанавливается точка с запятой (;) и указывается имя шаблона.

ПРИМЕЧАНИЕ

Шаблон должен быть определён перед его использованием в действии, иначе он игнорируется. Определения шаблонов всегда должны предшествовать определениям правил в **/etc/ rsyslog.conf**.



Любой вывод, который генерируется при помощи **rsyslog**, может быть изменён и отформатирован в соответствии с потребностями при помощи шаблонов (Templates). Для создания шаблона используется следующий синтаксис в конфигурационном файле **/etc/rsyslog.conf**:

```
template(name="TEMPLATE_NAME" type="string" string="text %PROPERTY% more text"
[option.OPTION="on"])
```

Где:

- **template()** – это директива представляющая блок определения шаблона.
- *TEMPLATE_NAME* – это обязательный аргумент, используемый для ссылки на шаблон. Все *TEMPLATE_NAME* должны быть уникальными.
- Обязательный аргумент type может принимать одно из следующих значений “**list**”, “**subtree**”, “**string**” или “**plugin**”.
- Аргумент **string** - это актуальный текст шаблона. Вместе с этим текстом могут использоваться специальные символы, такие как **\n** для перехода на новую строку или **\r** для возврата каретки. Прочие символы такие как **%** или ", должны быть исключены или экранированы для буквального использования.
- Текст, указанный между двумя знаками процента (**%**), указывает свойство (Property), которое должно разрешить доступ к указанному контенту сообщения **syslog**.
- Атрибут *OPTION* указывают любую опцию, которая изменяет в функционал шаблона (Template). На данный момент поддерживаются следующие опции шаблона **sql** и **stdsql**, которые используются для форматирования текста запроса SQL или **json**, который форматирует текст таким образом, чтобы он подходил для обработки JSON и **casesensitive**, который указывает чувствительность к регистру имён свойств.

ПРИМЕЧАНИЕ

Обратите внимание, что Database Writer проверяет указание опций **sql** или **stdsql** в шаблоне. Если они не указаны, писатель базы данных (Database Writer) не выполняет никаких действий. Это позволяет предотвратить любые возможные угрозы безопасности, такие как инъекции SQL.

Генерация динамических имен файлов.

Шаблоны могут быть использованы для генерации динамических имен файлов. При помощи указания свойства, как части пути файла, новый файл будет создан для каждого уникального свойства, что является удобным способом классификации сообщений **syslog**.

Пример использования свойства **timegenerated**, которое извлекает штамп времени из сообщения для генерации уникальных имен файлов каждого сообщения **syslog**:

```
template(name=”DynamicFile” type=”list”) {
constant(value=”/var/log/test_logs/”)
property(name=”timegenerated”)
constant(value”-test.log”)
}
```

Учтите, что директива **$template** только указывает шаблон. Она должна быть использована внутри правила, чтобы от нее был эффект. В **/etc/rsyslog.conf** используется знак вопроса (**?**) в определении действия для указания шаблона динамических имён файлов:

```
*.* ?DynamicFile
```

Свойства.

Свойства внутри шаблона (между двумя знаками процента (**%**)) позволяют получить различный контент сообщения syslog через заменитель свойств (Property Replacer). Для определения свойства внутри шаблона (между кавычек ("...")) используется следующий синтаксис:

```
%PROPERTY_NAME[:FROM_CHAR:TO_CHAR:OPTION]%
```

Где:

- Атрибут *PROPERTY_NAME* указывает имя свойства. Список всех доступных свойств и их подробное описание можно найти на странице руководства **rsyslog.conf(5)** в разделе Available Properties.
- Атрибуты *FROM_CHAR* и *TO_CHAR* обозначают диапазон символов, которые указывают свойство, на которое нужно реагировать. В качестве альтернативы можно использовать регулярные выражения для указания диапазона символов. Чтобы это сделать, необходимо указать букву R как атрибут *FROM_CHAR* и указать необходимое регулярное выражение как атрибут *TO_CHAR*.
- Атрибут *OPTION* указывает любую опцию свойства, такую как опция lowercase для конвертации ввода в нижний регистр. Список всех доступных опций свойств и их детальное описание можно на странице руководства **rsyslog.conf(5)** в секции Property Options.

Примеры простых свойств:

- Следующее свойство принимает весь текст сообщения **syslog**:

```
%msg%
```

- Следующее свойство принимает первые два символа текста сообщения **syslog**:

```
%msg:1:2%
```

- Следующее свойство принимает весь текст сообщения **syslog** и удаляет символ перехода на новую строку (Line Feed Character):

```
%msg:::drop-last-lf%
```

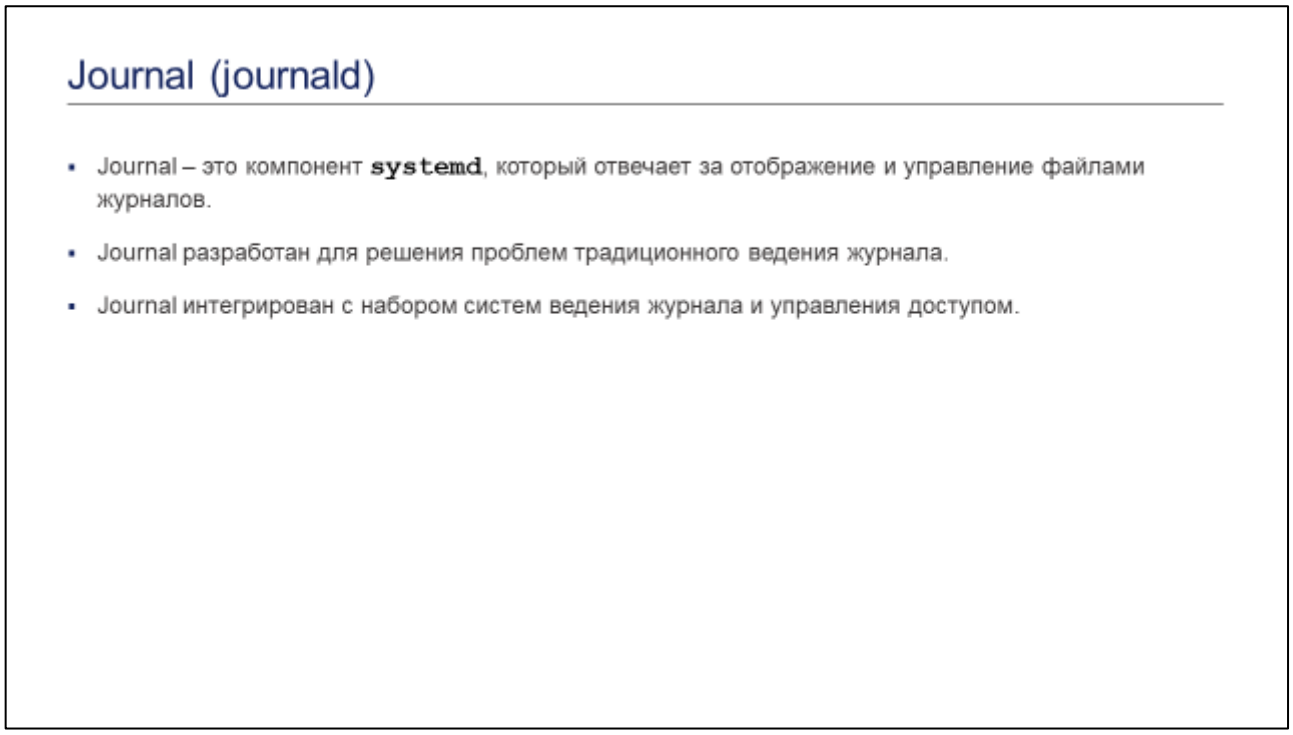
- Следующее сообщение принимает первые 10 символов штампа времени, который генерируется при получении сообщения **syslog**, и форматирует его в соответствии с стандартом даты RFC3999:

```
%timegenerated:1:10:date-rfc3339%
```

Глобальные директивы (Global Directives) – это опции настройки, которые применяются к демону **rsyslogd**. Они обычно показывают для специфических, определённых заранее переменных, которые влияют на поведение демона **rsyslogd** или последующие правила. Все глобальные директивы заключены в блок настройки **global**. Пример глобальной директивы, которая указывает перезапись имени локального хоста для сообщений журнала:

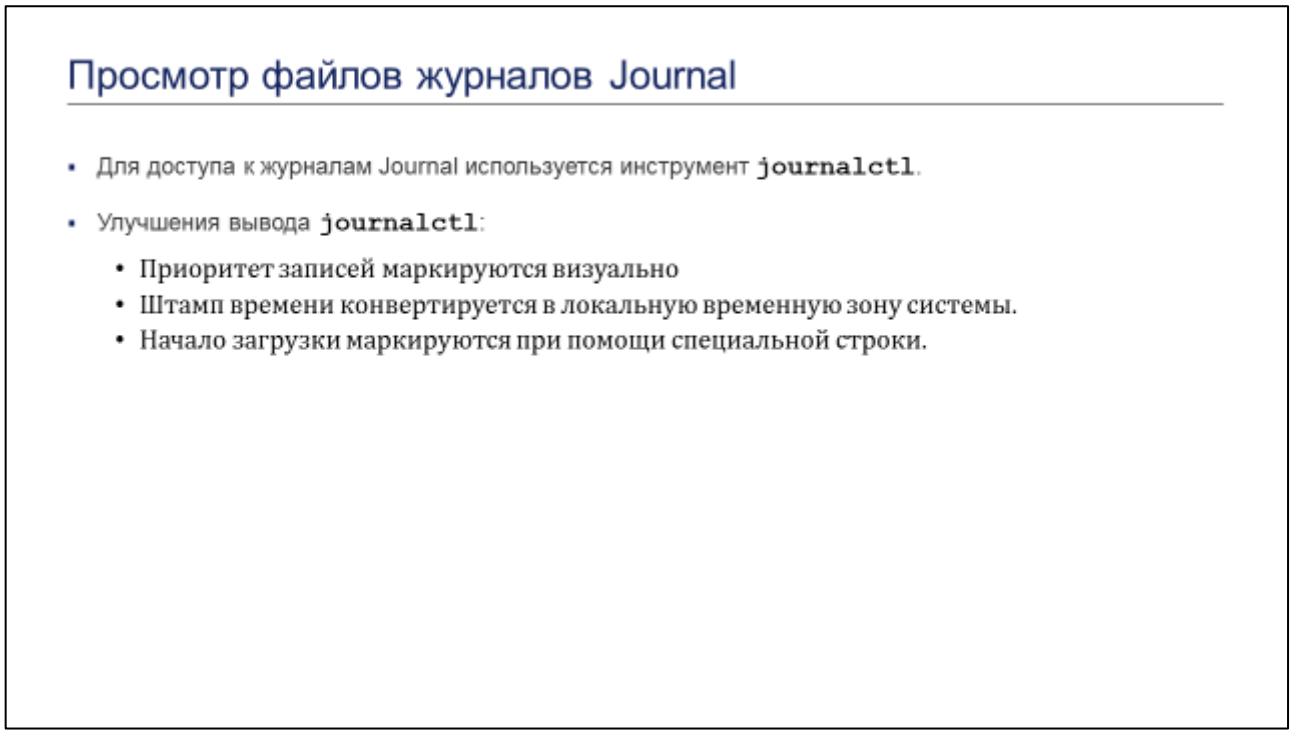
```
global(localHostname="machineXY")
```

Можно определить множество директив в конфигурационном файле **/etc/rsyslog.conf**. Директивы влияют на поведение всех опций настройки до тех пор, пока данная опция не указана еще раз с другим значением. Глобальный директивы могут быть использован для настройки действий, очередей и для отладки.



Journal – это компонент **systemd**, который отвечает за отображение и управление файлами журналов. Он может быть использован совместно или вместо традиционного демона syslog, такого как **rsyslogd**. Journal разработан для решения проблем, связанных традиционным ведением журнала. Он плотно интегрирован с набором систем, поддерживающих различные технологии ведения журнала и управления доступами к файлам журнала.

Данные ведения журнала накапливаются, хранятся и обрабатываются при помощи сервиса Journal – **jounrald**. Он создает и поддерживает бинарные файлы называемые журналы (Journals) на базе информации ведения журнала, которую получает от ядра, пользовательских процессов, со стандартного вывода и стандартного вывода ошибок системных сервисов или через стандартный API. Данные журналы структурированы и индексированы, что обеспечивает относительно высокую скорость поиска. Записи Journal могут обладать уникальным идентификатором. Сервис **journald** собирает набор полей метаданных для каждого сообщения журнала. Актуальные файлы журнала защищены и поэтому не могут быть отредактированы вручную.



Для доступа к журналам Journal используется инструмент **journalctl**. Для базового просмотра журналов необходимо ввести следующую команду от имени root:

```
# journalctl
```

Вывод данной команды – список всех сгенерированных файлов журналов на системе в том числе сообщений сгенерированных системными компонентами и пользователями. Структура вывода похожа на ту, которая используется в **/var/log/messages** но с важными улучшениями:

- Приоритет записей маркируются визуально. строки с приоритетом ошибка (Error) и выше выделяются красным цветом, а строки с приоритетом предупреждение (Warning) выделяются жирным шрифтом.
- Штамп времени конвертируется в локальную временную зону системы.
- Все данные, попавшие в журнал, отображаются, в том числе из журналов, прошедших ротацию.
- Начало загрузки маркируются при помощи специальной строки.

Пример вывода **journalctl**:

```
# journalctl
-- Logs begin at Sat 2018-09-22 10:32:35 +06, end at Mon 2018-09-24 15:34:40 +06
Sep 22 10:32:35 mainserver01 systemd-journal[85]: Runtime journal is using 4.0M
Sep 22 10:32:35 mainserver01 kernel: Initializing cgroup subsys cpuset
Sep 22 10:32:35 mainserver01 kernel: Initializing cgroup subsys cpu
Sep 22 10:32:35 mainserver01 kernel: Initializing cgroup subsys cpuacct
Sep 22 10:32:35 mainserver01 kernel: Linux version 3.10.0-862.11.6.el7.x86_64 (b
Sep 22 10:32:35 mainserver01 kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-86
lines 1-7
```

Во многих случаях, необходимо только несколько последних записей из журналов. Простейший способ сократить вывод **journalctl** это использовать опцию -n которая отображает только указанное число из самых последних записей:

```
# journalctl -n 10
```

Команда **journalctl** позволяет контролировать форму вывода при помощи следующего синтаксиса:

```
# journalctl -o FORM
```

Замените FORM на ключевое слово, указывающее необходимую форму вывода. Существует несколько опций, таких как **verbose**, который возвращает полностью структурированные элементы записей со всеми полями; **export**, который создается бинарный поточный файл, подходящий для резервного копирования и передачи по сети; а также **json**, который форматирует записи формате структуры данных JSON. Полный список поддерживаемых ключевых слов доступен на странице руководства **journalctl(1)**.

Пример подробного вывода **journalctl**:

```
# journalctl -o verbose -n 1
-- Logs begin at Sat 2018-09-22 10:32:35 +06, end at Wed 2018-09-26 08:15:00 +06
Wed 2018-09-26 08:15:00.087333 +06 [s=32f351cbd8634635bb020d0cb7f5337d;i=2bab;b=
    _TRANSPORT=syslog
    PRIORITY=6
    _SYSTEMD_SLICE=system.slice
    _BOOT_ID=2cfe7201072942ce954f1b055142f78b
    _MACHINE_ID=880be116fd0547f5b1b46e01c0c8e4c2
    _HOSTNAME=mainserver01
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=named
    SYSLOG_PID=947
    _PID=947
    _UID=25
    _GID=25
    _COMM=named
    _EXE=/usr/sbin/named
    _CMDLINE=/usr/sbin/named -u named -c /etc/named.conf
    _CAP_EFFECTIVE=1000400
    _SYSTEMD_CGROUP=/system.slice/named.service
    _SYSTEMD_UNIT=named.service
    MESSAGE=error (network unreachable) resolving 'wd-prod-ss-eu-west-2-fe.weste
    _SOURCE_REALTIME_TIMESTAMP=1537928100087333
lines 1-22/22 (END)
```

Полное описание всех возможных полей доступно на странице руководства **systemd.journal-fields(7)**.

Управление доступом.

По умолчанию пользователи Journal без привилегий root могут просматривать только сгенерированные самим пользователем файлы журналов. Для того чтобы предоставить выбранным права доступа ко всем файлам журналов необходимо их добавить в группу **adm**.

```
# usermod -a -G adm ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

Управление доступом к Journal работает только в том случае если включена постоянное хранение (Persistent Storage).

Использование живого просмотра (Live View).

При вызове **journalctl** без параметров, отображается полный список сообщений журнала, начинающийся с самых старых собранных записей. При помощи живого просмотра (Live View) можно отслеживать сообщения журнала в реальном времени по мере их добавление в журнал. Для запуска **journalctl** в режиме живого просмотра (Live View) введите следующую команду:

```
# journalctl -f
```

Вывод команды **journalclt** без параметров очень большой, поэтому присутствует возможность использовать различные методы фильтрации для извлечения необходимой информации.

Фильтрация по приоритету.

Сообщения журнала часто используются для отслеживания ошибочного поведение на системе. Для просмотра записей с указанным приоритетом и выше используется следующий синтаксис:

```
# journalctl -p PRIORITY
```

необходимо заменить *PRIORITY* при помощи одного из ключевых слов (или при помощи цифры): **debug** (**7**), **info** (**6**), **notice** (**5**), **warning** (**4**), **err** (**3**), **crit** (**2**), **alert** (**1**), **emerg** (**0**).

Фильтрация по времени.

Для просмотра записей журнала, начиная с текущей загрузки, используется ключ **-b**:

```
# journalctl -b
```

Если перезагрузка прошла случайно, то **journalctl** не сократит вывод. В таком случае используется фильтрация на базе времени:

```
# journalctl --since=value --until=value
```

При помощи ключей **--since** и **--until** можно просмотреть сообщение журнала только в рамках указанного диапазона. Можно передавать значения в формате даты, времени или обоих, как показано в примере ниже:

```
# journalctl -p warning --since="2018-9-16 23:59:59"
```

Расширенная фильтрация.

Пример подробного вывода **journalctl**:

```
# journalctl -o verbose -n 1
-- Logs begin at Sat 2018-09-22 10:32:35 +06, end at Wed 2018-09-26 08:15:00 +06
Wed 2018-09-26 08:15:00.087333 +06 [s=32f351cbd8634635bb020d0cb7f5337d;i=2bab;b=
    _TRANSPORT=syslog
    PRIORITY=6
    _SYSTEMD_SLICE=system.slice
    _BOOT_ID=2cfe7201072942ce954f1b055142f78b
    _MACHINE_ID=880be116fd0547f5b1b46e01c0c8e4c2
    _HOSTNAME=mainserver01
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=named
    SYSLOG_PID=947
    _PID=947
    _UID=25
    _GID=25
    _COMM=named
    _EXE=/usr/sbin/named
    _CMDLINE=/usr/sbin/named -u named -c /etc/named.conf
    _CAP_EFFECTIVE=1000400
    _SYSTEMD_CGROUP=/system.slice/named.service
    _SYSTEMD_UNIT=named.service
    MESSAGE=error (network unreachable) resolving 'wd-prod-ss-eu-west-2-fe.weste
    _SOURCE_REALTIME_TIMESTAMP=1537928100087333
lines 1-22/22 (END)
```

Как показано в примере подробного вывода, он выводит набор полей указанной записи журнала, по этим полям можно выполнить фильтрацию. Для получения полного описания метаданных который может хранить **systemd** обратитесь к странице руководства **systemd.journal-fields(7)**. Эти метаданные собираются для каждого сообщения журнала без участия пользователя. Значения, как правило текстовые, но также могут быть бинарными и достаточно большими, также поля могут иметь несколько значений, хотя это не очень распространено.

Для просмотра списка уникальных значений, которые хранится в указанном поле, используется следующий синтаксис:

```
# journalctl -F FIELDNAME
```

Для отображения записей журнала, которые содержат определенное значение в указанном поле, используется следующий синтаксис:

```
# journalctl FIELDNAME=VALUE
```

ПРИМЕЧАНИЕ

Так как число полей метаданных, хранимых в **systemd**, достаточно большое, можно попросту забыть точное имя необходимого поля. В таком случае необходимо ввести команду **journalctl** и дважды нажать клавишу **<Tab>**:

```
# journalctl
```

Аналогично, для получения списка уникальных значений для поля, можно ввести имя поля, поставить знак равно (**=**) и дважды нажать **<Tab>**:

```
# journalctl _HOSTNAME=
```

Можно указать множество значений для одного поля:

```
# journalctl FIELDNAME=VALUE1 FIELDNAME=VALUE2 ...
```

Указание двух значений для одного поля обрабатывается как логическое комбинация ИЛИ и записи с полем, соответствующим первому или второму значению отображаются.

Также есть возможность указать несколько полей для сокращения в выводимого набора:

```
# journalctl FIELDNAME1=VALUE FIELDNAME2=VALUE ...
```

Если указаны два разных поля они будут объединяться при помощи логического И, в результате будут отображаться только те записи у которых оба указанных поля соответствуют указанным значениям.

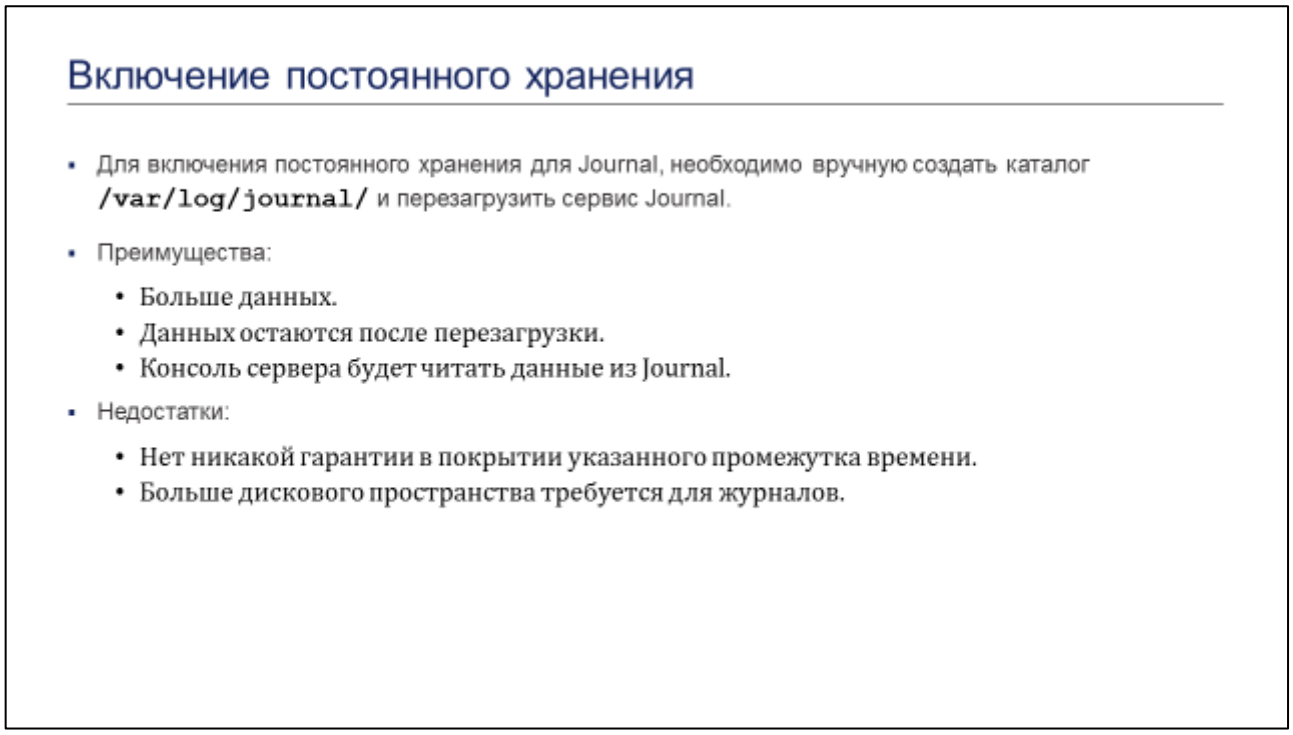
Для того чтобы заменить логическое и на логическое или используется знак плюс (**+**):

```
# journalctl FIELDNAME1=VALUE + FIELDNAME2=VALUE ...
```

Данная команда возвращает записи, которые соответствуют хотя бы одному из указанных условий.

Любые фильтры можно применять к режиму живого просмотра (Live View), для отображения только в выбранной группы записи журнала:

```
# journalctl -f FIELDNAME=VALUE ...
```



По умолчанию Journal хранит файлы журналов только в памяти или маленьком циклическом буфере в каталоге **/run/log/journal/**. Данный виртуальный каталог является энергозависимым и данные журналов не сохраняются после перезагрузки. При помощи конфигурация по умолчанию, **syslog** читает журналы Journal и сохраняет их в каталоге **/var/log/**. При помощи включения постоянного хранения, файлы Journal сохраняются в каталоге **/var/log/journal**, это значит, что они останутся после перезагрузки.

Включение постоянного хранения имеет следующие преимущества:

- Больше данных записывается для устранения неисправностей на более продолжительный период времени.
- Для немедленного устранения неисправностей, больше данных остаются доступными после перезагрузки.
- Консоль сервера будет читать данные из Journal, а не из файлов журнала.

Постоянное хранение также имеет некоторые недостатки:

- Даже при включенном постоянном хранении, объём хранимых данных зависит от свободной памяти, нет никакой гарантии в покрытии указанного промежутка времени.
- Больше дискового пространства требуется для журналов.

Для включения постоянного хранения для Journal, необходимо вручную создать каталог **/var/log/journal/**:

```
# mkdir -p /var/log/journal/
```

После этого необходимо перезагрузить сервис Journal для применения изменений:

```
# systemctl restart systemd-journald
```

SSH (Secure Shell) – это протокол, который обеспечивает безопасные коммуникации между двумя системами при помощи клиент-серверной архитектуры, а также позволяет пользователям удаленно входить на хостовые системы. В отличии от прочих протоколов удаленного взаимодействия, таких как FTP или telnet, SSH шифрует сессию входа, тем самым делая затруднительным перехват незашифрованного пароля.

Программа **ssh** разработана для замены более старых и менее безопасных терминальных приложений, используемых для входа на удаленную машину, таких как **telnet** и **rsh**. Связанная программа **scp** заменяет более старую программу **rcp**, разработанную для копирования файлов между хостами. В связи с тем, что эти старые программы не шифруют передачу пароля между клиентом и сервером, постарайтесь, по возможности, не использовать их. Использование безопасных методов входа на удаленные системы, снижает риски для клиентской и удаленной систем.

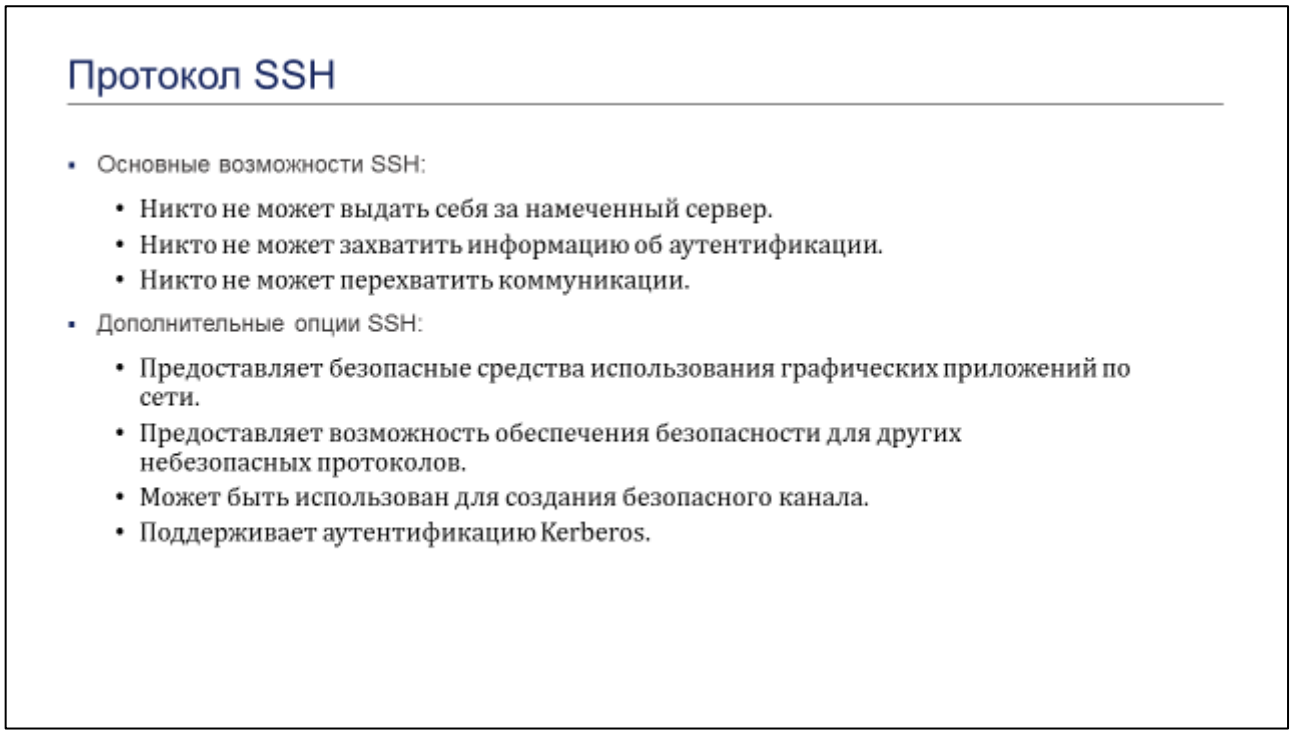
RedOS включает основной пакет OpenSSH (**openssh**), а также пакеты сервера OpenSSH (**openssh-server**) и клиента (**openssh-clients**). Обратите внимание, что пакеты OpenSSH требуют пакет OpenSSL (**openssl-libs**), который устанавливает несколько важных криптографических библиотек, позволяя OpenSSH предоставлять зашифрованные коммуникации.

Версии протокола.

Две версии протокола SSH существуют на данный момент: версия 1 и более новая версия 2. Комплект OpenSSH, входящий в состав RedOS, использует SSH версии 2, который использует улучшенный алгоритм обмена ключами, не подверженный известным уязвимостям первой версии. Однако, в целях обеспечения совместимости комплект OpenSSH также обеспечивает поддержку для соединений первой версии.

ПРИМЕЧАНИЕ

Для обеспечения максимальной безопасности ваших подключений, рекомендуется использовать клиенты и серверы совместимые только со второй версией протокола SSH.



Потенциальные злоумышленники обладают различными инструментами для перехвата, подмены и перенаправления траффика в целях получения доступа к системе. В общем виде эти угрозы могут быть категорированы следующим образом:

Перехват коммуникаций между двумя хостами.

Злоумышленник может располагаться где угодно в сети, между взаимодействующими партнерами и копировать любую информацию, передаваемую между ними. Он может перехватывать и хранить ее или изменять и отправлять ее ожидающему получателю.

Подобная атака обычно выполняется при помощи сниффера пакетов (Packet Sniffer), это может быть практически любая сетевая утилита, способная захватывать каждый пакет, идущий через сеть и анализировать его содержимое.

Подмена одного из хостов.

Система злоумышленника может быть сконфигурирована таким образом, что будет представляться как получатель передачи. Если это сработает, то пользовательская система не узнает, что она взаимодействует с подставным хостом.

Подобная атака может быть выполнена через отравление DNS (DNS Poisoning) или через имитацию IP (IP Spoofing). В первом случае злоумышленник использует взломанный сервер DNS для перенаправления клиентских систем на дублирующий вредоносный хост. Во втором случае злоумышленник посылает фальшивые сетевые пакеты, как будто они отправлены с доверенного хоста.

Обе техники направлены на перехват чувствительной информацию и, в случае успеха злоумышленника, последствия могут быть катастрофическими. Если используется SSH для удаленного входа в оболочку и копирования файлов эти угрозы безопасности будут существенно снижены. Это связано с тем, что клиент и сервер SSH используют цифровые подписи для проверки идентичности. Дополнительно, все коммуникации меду клиентом и сервером шифруются.

Попытка подмены идентификатора на любой стороне не сработает, так как каждый пакет шифруется ключами, известными только на локальной и удаленной системе.

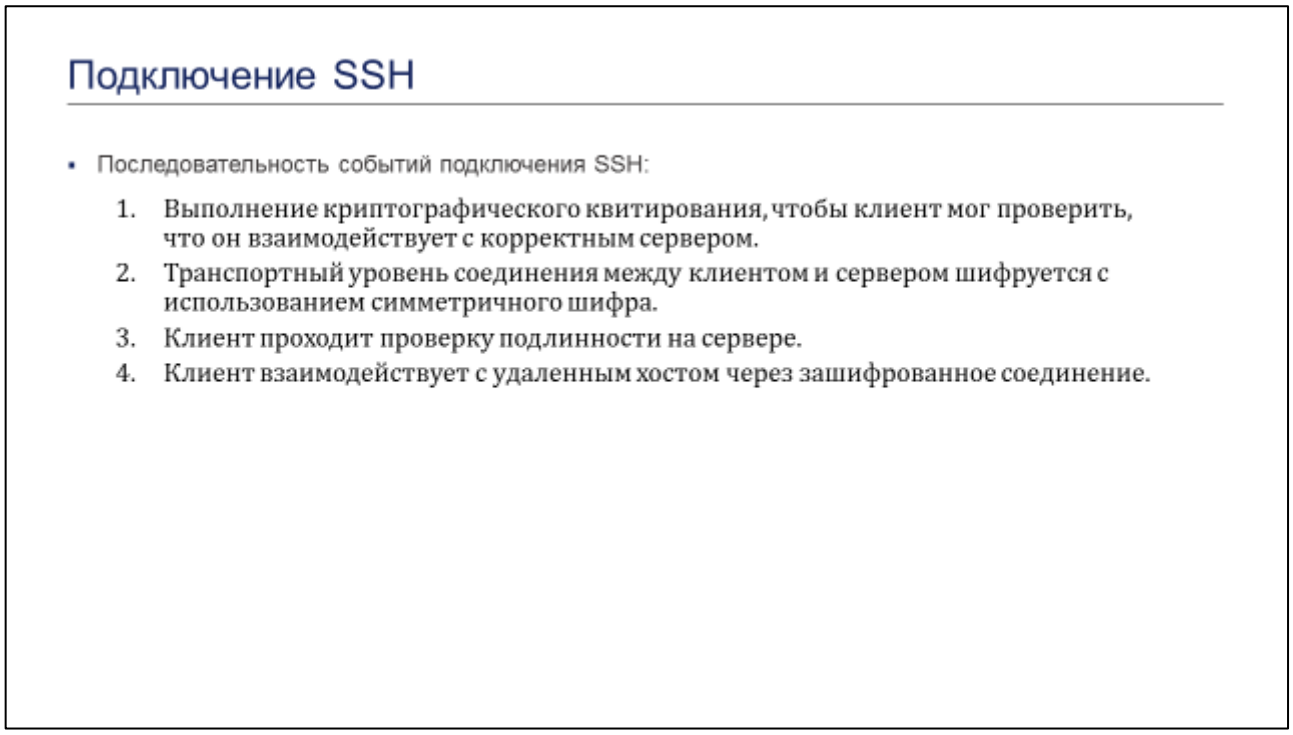
Основные возможности.

Протокол SSH обеспечивает следующее:

- Никто не может выдать себя за намеченный сервер. После начального подключения клиент может удостовериться, что подключается к тому же серверу, с которым соединялся в прошлый раз.
- Никто не может захватить информацию об аутентификации. Клиент передает информацию по аутентификации на сервер с использованием 128-битного шифрования.
- Никто не может перехватить коммуникации. Все данные отправленные и полученные в рамках сессии, передаются с использованием 128-битного шифрования, делая крайне сложным расшифровку и чтение перехваченной информации.

Дополнительно SSH предлагает следующие опции:

- Предоставляет безопасные средства использования графических приложений по сети. При помощи технологии перенаправления X11 (X11 Forwarding) приложений с сервера.
- Предоставляет возможность обеспечения безопасности для других небезопасных протоколов. Протокол SSH шифрует все что он отправляет и получает. При помощи техники перенаправления порта (Port Forwarding) сервер SSH может обеспечить безопасный канал для небезопасных протоколов, таких как POP, и увеличить безопасности системы и данных.
- Может быть использован для создания безопасного канала. Сервер и клиент OpenSSH могут быт настроены на создание тоннеля, подобному виртуальной частной сети (VPN), для траффика между серверной и клиентской машинами.
- Поддерживает аутентификацию Kerberos. Серверы и клиенты OpenSSH могут быть настроены на аутентификацию с использованием GSSAPI (Generic Security Services Application Program Interface) внедрения сетевого протокола проверки подлинности Kerberos.



Следующая серия событий помогает защитить целостность коммуникаций SSH между двумя хостами:

1. Выполнение криптографического квитирования, чтобы клиент мог проверить, что он взаимодействует с корректным сервером.
2. Транспортный уровень соединения между клиентом и сервером шифруется с использованием симметричного шифра.
3. Клиент проходит проверку подлинности на сервере.
4. Клиент взаимодействует с удаленным хостом через зашифрованное соединение.

Транспортный уровень.

Основная роль транспортного уровня – это упрощение безопасного и защищенного взаимодействия между двумя хостами в процессе аутентификации и последующего взаимодействия. Транспортный уровень выполняет это за счет шифрования и расшифровывания данных и обеспечения защиты целостности пакетов данных в процессе отправки и получения. Транспортный уровень также предоставляет сжатие, ускоряю передачу информации.

Когда клиент SSH контактирует с сервером, происходит обмен информацией ключей, таким образом, чтобы две системы могли корректно построить транспортный уровень. Следующие шаги происходят во время обмена:

- Обмен ключами.
- Определение алгоритма шифрования публичны ключом.
- Определение алгоритма симметричного шифрования.
- Определение алгоритма аутентификации сообщений.
- Определение алгоритма хеширования.

В процессе обмена ключами, сервер определяет себя для клиента при помощи уникального ключа хоста (Host Key). Если клиент ранее не взаимодействовал с данным сервером, ключ хоста сервера

не известен клиенту, и он не может соединиться. OpenSSH решает эту проблему принимая ключ хоста сервера. Это выполняется после уведомления пользователя с последующем принятием и проверки нового ключа хоста. При последующих подключениях, ключ хоста сервера будет проверятся с версией, сохраненной на клиенте, тем самым гарантируя, что клиент соединяется с тем же самым сервером. Если в будущем ключ хоста не совпадет, пользователь должен будет удалить сохраненную версию ключа, чтобы установить соединение.

ПРЕДУПРЕЖДЕНИЕ

Это возможно для злоумышленника, замаскировать сервер SSH во время первой сессии, когда локальная система не знает различий между настоящим сервером и подставным. Чтобы избежать этого, убедитесь в корректности подключения к хосту, взаимодействуя с администратором сервера, при первом подключении к серверу.

SSH разработан с возможностью работы с поддержкой любых алгоритмов с использованием крытого ключа и форматов шифрования. После того как начальный обмен ключами создает значение хеш-функции, используемое для обмена значением общего секрета, две системы немедленно приступают к вычислению новых ключей и алгоритмов для защиты аутентификации, и будущей передачи данных через подключение.

После передачи определенного объема данных с использованием определенного ключа и алгоритма (точный объем данных зависит от реализации SSH), еще один обмен ключами происходит, генерируется новый набор значений хеш-функций и значения общего секрета, данная информация будет применима на ограниченный период времени.

Аутентификация.

После построения транспортного уровня, безопасный туннель передает информацию между двумя системами, сервер сообщает клиенту различные поддерживаемые методы аутентификации, такие как использование шифрования подписи частным ключом или ввод пароля. Клиент пытается аутентифицировать на сервере с использованием одного из поддерживаемых методов.

SSH серверы и клиенты могут быть настроены для разрешения различных типов аутентификации, которые позволяют каждой стороне оптимальный объем контроля. Сервер может выбрать какой метод шифрования он поддерживает на базе собственной модели безопасности, и клиент может выбрать порядок методов аутентификации исходя из доступных опций.

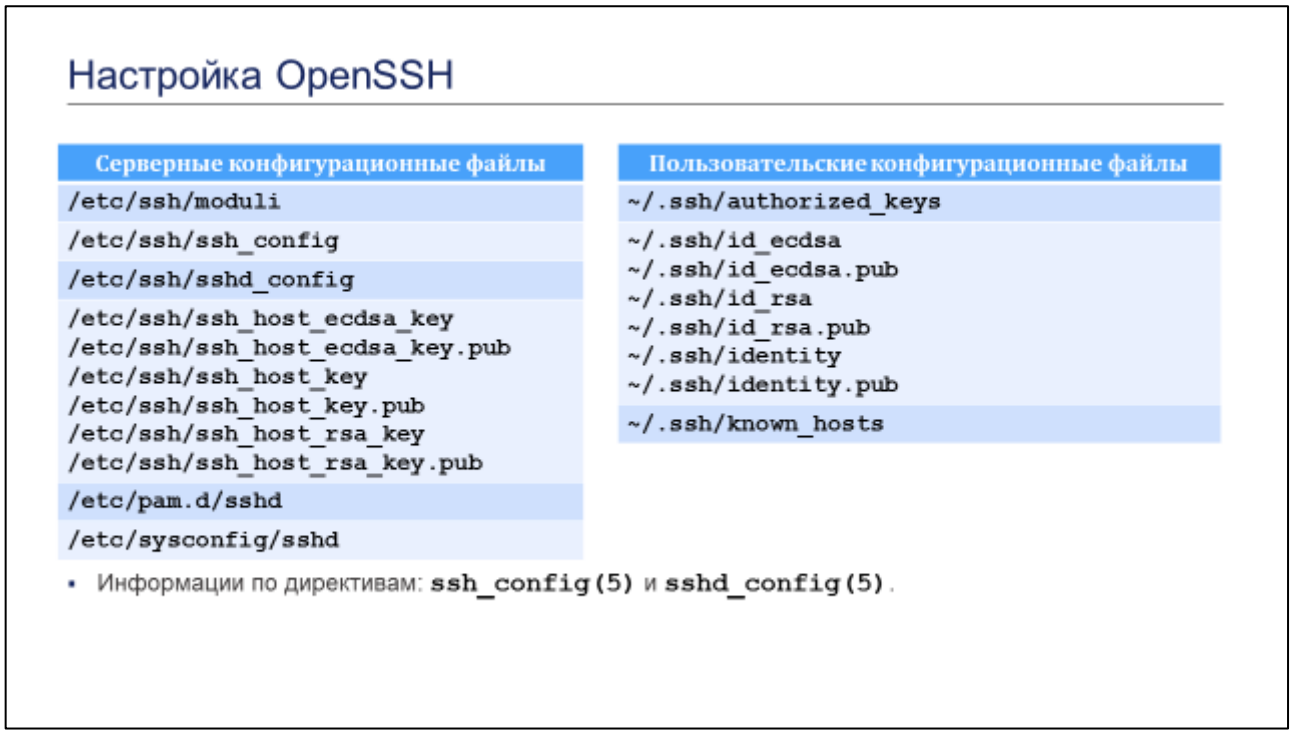
Каналы.

После успешной аутентификации через транспортный уровень SSH, несколько каналов открывается через технику, называемую мультиплексирование (Multiplexing). Каждый из этих каналов обрабатывает коммуникации для различных терминальных сессий и для перенаправления сессий X11.

Клиенты и серверы могут создавать новые каналы. Затем каждый канал назначает разный номер на каждый конец соединения. Когда клиент пытается открыть новый канал. Клиенты отправляют номер канала вместе с запросом. Это информация сохраняется на сервере и используется для прямого взаимодействия с каналом. Это делается для того, чтобы различные типы сессии не влияли друг на друга и чтобы, когда одна из сессий завершается, ее канал был закрыт без уничтожения основного соединения SSH.

Также каналы поддерживают управление потоком (Flow-Control), которое позволяет ему отправлять и принимать данные в организованном виде. В этом случае, данные не отправляются через канал до тех пор, пока клиент не получит сообщение, что этот канал открыт.

Клиент и сервер обсуждают характеристики каждого канала автоматически, в зависимости от типа сервиса, запрошенного клиентом и пути, по которому пользователь подключается к сети. Это обеспечивает потрясающую гибкость в обработке различных типов удаленных подключений без необходимости изменять базовую инфраструктуру протокола.



В OpenSSH существует два различных набора конфигурационных файлов: набор для клиентских программ (таких как **ssh**, **scp** и **sftp**) и набор для сервера (демона **sshd**).

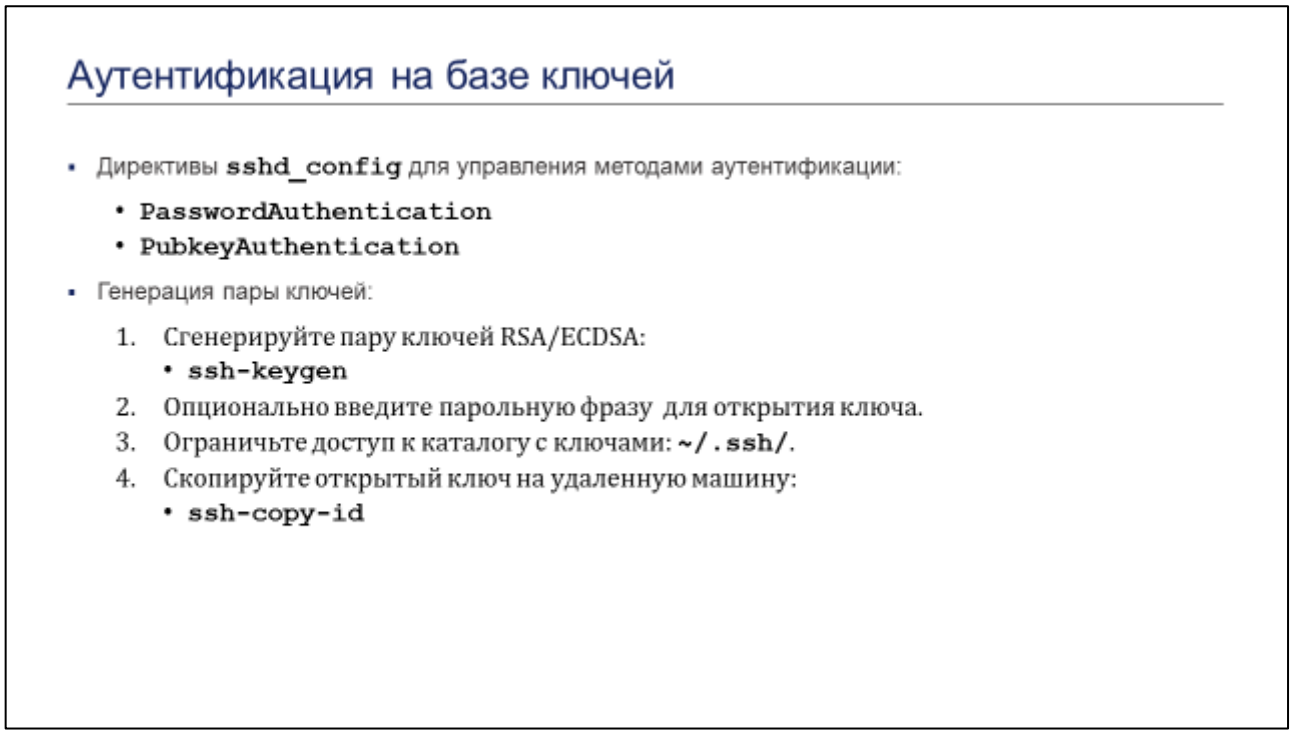
Общая системная информация о конфигурации SSH хранится в каталоге **/etc/ssh/**, как описано ниже:

- **/etc/ssh/moduli** – Содержит группы Диффи-Хеллмана (Diffie-Hellman), используемые для обмена ключами Диффи-Хеллмана, что является критичным для построения безопасного транспортного уровня. Когда обмен ключами совершен в начале сессии SSH, создается значение общего секрета. Это значение в дальнейшем используется для обеспечения аутентификации хоста.
- **/etc/ssh/ssh_config** – Конфигурационный файл клиента SSH по умолчанию. Обратите внимание, что он переписывается файлом **~/.ssh/config**, если последний существует.
- **/etc/ssh/sshd_config** – Конфигурационный файл демона **sshd**.
- **/etc/ssh/ssh_host_ecdsa_key** – Частный ключ ECDSA, используемый демоном **sshd**.
- **/etc/ssh/ssh_host_ecdsa_key.pub** – Открытый ключ ECDSA, используемый демоном **sshd**.
- **/etc/ssh/ssh_host_key** – Частный ключ RSA, используемый демоном sshd для протокола SSH версии 1.
- **/etc/ssh/ssh_host_key.pub** – Открытый ключ RSA, используемый демоном **sshd** для протокола SSH версии 1.
- **/etc/ssh/ssh_host_rsa_key** - Частный ключ RSA, используемый демоном sshd для протокола SSH версии 2.
- **/etc/ssh/ssh_host_rsa_key.pub** - Открытый ключ RSA, используемый демоном sshd для протокола SSH версии 2.
- **/etc/pam.d/sshd** – Конфигурационный файл PAM, для демона **sshd**.
- **/etc/sysconfig/sshd** – Конфигурационный файл для службы **sshd**.

Пользовательская информация о конфигурации SSH хранится в каталоге ~/.ssh/ домашнего каталога пользователя, как описано в ниже:

- **~/.ssh/authorized_keys** – Удерживает список авторизованных публичных ключей для серверов. Когда клиент подключается к серверу, сервер аутентифицирует клиент, за счет проверки публичного ключа, хранящегося в данном файле.
- **~/.ssh/id_ecdsa** – Содержит частный ключ ECDSA для пользователя.
- **~/.ssh/id_ecdsa.pub** – Содержит открытый ключ ECDSA для пользователя.
- **~/.ssh/id_rsa** – Содержит частный ключ RSA, используемый ssh для протокола SSH версии 2.
- **~/.ssh/id_rsa.pub** – Содержит открытый ключ RSA, используемый ssh для протокола SSH версии 2.
- **~/.ssh/identity** – Содержит частный ключ RSA, используемый ssh для протокола SSH версии 1.
- **~/.ssh/identity.pub** - Содержит открытый ключ RSA, используемый ssh для протокола SSH версии 1.
- **~/.ssh/known_hosts** – Содержит ключи хостов от серверов SSH к которым подключался пользователь. Этот файл очень важен для подтверждения того, что клиент SSH подключается к корректному серверу SSH.

Для получения информации по различным директивам, которые могут быть использованы в конфигурационных файлах SSH, обратитесь к страницам руководства: **ssh_config(5)** и **sshd_config(5)**.



Для увеличения безопасности в будущем, сгенерируйте пару ключей SSH и примените аутентификацию на базе ключей, отключив аутентификацию на базе паролей. Чтобы это сделать, откройте конфигурационный файл **/etc/ssh/sshd_config** в текстовом редакторе и измените опцию **PasswordAuthentication** следующим образом:

```
PasswordAuthentication no
```

Если вы работаете на системе, которая не была только что установлена, проверьте, что не установлена опция **PubkeyAuthentication no**. Если вы подключаетесь удаленно, перед отключением аутентификации по паролю, протестируйте возможность использования аутентификации на базе ключей.

Чтобы иметь возможность использовать **ssh**, **scp** или **sftp** для подключения к серверу с клиентской машины, сгенерируйте пару ключей авторизации, как описано в процедуре ниже. Обратите внимание, что ключи должны быть сгенерированы для каждого пользователя отдельно.

RedOS 7 использует протокол SSH версии 2 по умолчанию.

ПРИМЕЧАНИЕ

Если вы выполните процедуру от имени пользователя root, только root, только root будет иметь возможность использовать ключи.

ПРИМЕЧАНИЕ

Если вы переустанавливаете вашу систему и хотите сохранить сгенерированную ранее пару ключей, сделайте резервную копию каталога **~/.ssh/**. После переустановки, скопируйте каталог обратно в домашнюю папку. Это процедура может быть выполнена для всех пользователей системы, включая пользователя root.

Генерация пары ключей.

Для генерации пары ключей RSA для протокола SSH версии 2, воспользуйтесь следующими шагами:

1. Сгенерируйте пару ключей RSA, при помощи следующей команды:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa):
```

2. Нажмите **<Enter>** для подтверждения расположения по умолчанию (**~/.ssh/id_rsa**), для вновь созданного ключа.
3. Введите парольную фразу и подтвердите ее, когда она будет запрошена вновь. Для обеспечения безопасности не используйте пароль от учетной записи. После этого вам будет выдано следующее сообщение:

```
Your identification has been saved in /home/Student/.ssh/id_rsa.
Your public key has been saved in /home/Student/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14
student@c7-client01.ll-100.local
The key's randomart image is:
+--[ RSA 2048]----+
|              E.  |
|             . .  |
|              o . |
|               . .|
|        S .    . |
|         + o o ..|
|          * * +oo|
|           O +..=|
|           o*  o.|
+-----------------+
```

4. По умолчанию права на каталог **~/.ssh/** установлены в **rwx------** или в восьмеричной системе счисления **700**. Это делается для того, чтобы только пользователь student мог просматривать содержимое. Чтобы в этом убедиться выполните следующую команду:

```
$ ls -ld ~/.ssh
drwx------. 2 student student 54 Nov 25 16:56 /home/Student/.ssh/
```

5. Чтобы скопировать открытый ключ на удаленную машину, выполните следующую команду:

```
$ ssh-copy-id user@hostname
```

Эта команда копирует последний измененный открытый ключ **~/.ssh/id*.pub**, если он еще не установлен. В качестве альтернативы можно указать имя файла следующим образом:

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

Эта команда скопирует содержимое файла **~/.ssh/id_rsa.pub** в файл **~/.ssh/authorized_keys** на машине к которой вы хотите подключиться. Если файл уже существует, ключ дописывается в конец.

Для генерации пары ключей ECDSA для протокола SSH версии 2, воспользуйтесь следующими шагами:

1. Сгенерируйте пару ключей ECDSA при помощи следующей команды:

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_ecdsa):
```

2. Нажмите **<Enter>** для подтверждения расположения по умолчанию (**~/.ssh/id_ecdsa**), для вновь созданного ключа.
3. Введите парольную фразу и подтвердите ее, когда она будет запрошена вновь. Для обеспечения безопасности не используйте пароль от учетной записи. После этого вам будет выдано следующее сообщение:

```
Your identification has been saved in /home/Student/.ssh/id_ecdsa.
Your public key has been saved in /home/student/.ssh/id_ecdsa.pub.
The key fingerprint is:
fd:1d:ca:10:52:96:21:43:7e:bd:4c:fc:5b:35:6b:63
Student@c7-client01.ll-100.local
The key's randomart image is:
+--[ECDSA 256]---+
|       .+ +o     |
|       . =.o     |
|        o o + ..|
|         + + o +|
|       S o o oE.|
|          + oo+.|
|           + o  |
|                |
|                |
+-----------------+
```

4. По умолчанию права на каталог **~/.ssh/** установлены в **rwx------** или в восьмеричной системе счисления **700**. Это делается для того, чтобы только пользователь student мог просматривать содержимое. Чтобы в этом убедиться выполните следующую команду:

```
$ ls -ld ~/.ssh
drwx------. 2 Student Student 54 Nov 25 16:56 /home/student/.ssh/
```

5. Чтобы скопировать открытый ключ на удаленную машину, выполните следующую команду:

```
$ ssh-copy-id user@hostname
```

Эта команда копирует последний измененный открытый ключ **~/.ssh/id*.pub**, если он еще не установлен. В качестве альтернативы можно указать имя файла следующим образом:

```
$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub user@hostname
```

Эта команда скопирует содержимое файла **~/.ssh/id_ecdsa.pub** в файл **~/.ssh/authorized_keys** на машине к которой вы хотите подключиться. Если файл уже существует, ключ дописывается в конец.

ПРИМЕЧАНИЕ

Частный ключ предназначен только для персонального использования пользователем, который его сгенерировал, поэтому очень важно никогда не передавать его кому-либо.

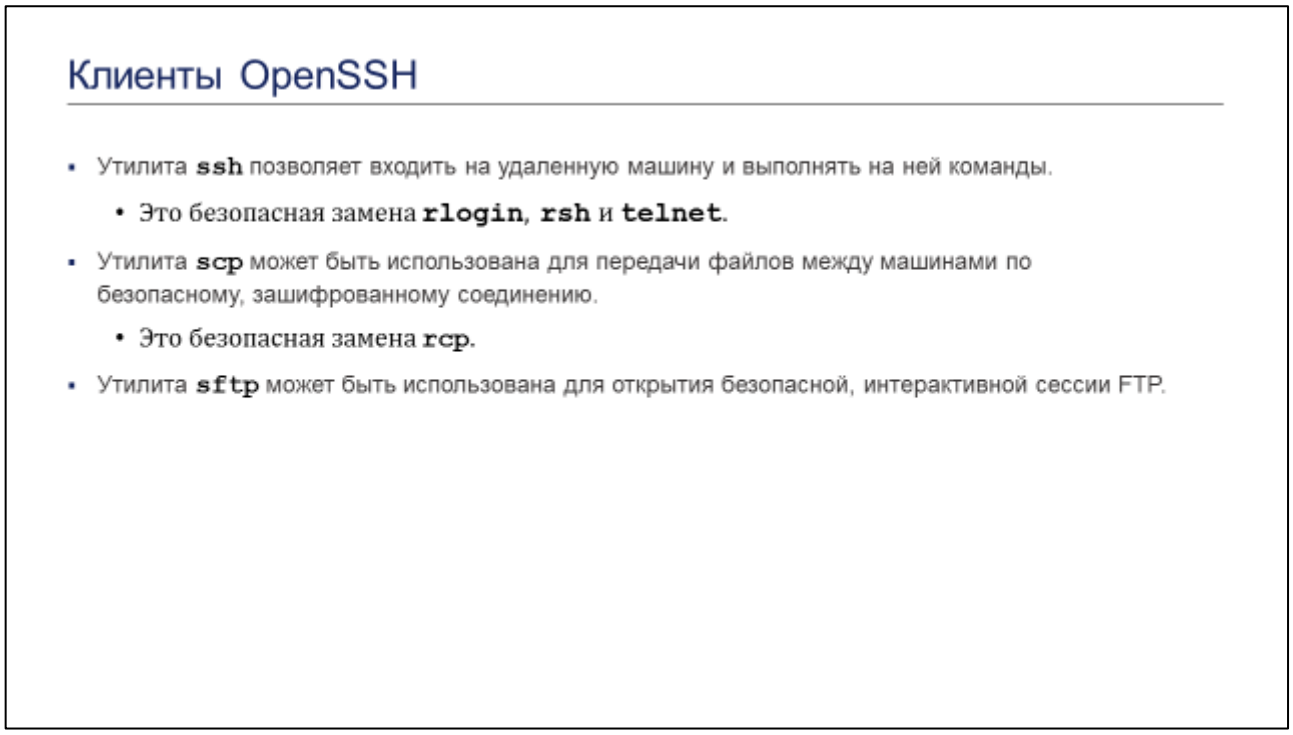
Настройка ssh-agent.

Для того чтобы сохранить парольную фразу и не вводить ее каждый раз при инициализации подключения с удаленной машиной, вы можете использовать аутентификацию агента **ssh-agent**. В рамках графической сессии это можно сделать при помощи инструмента Stareup Applications Preferences (потребуется установленный пакет **openssh-askpath**).

Для сохранения парольной фразы для оболочки командной строки, используйте следующую команду:

```
$ ssh-add
Enter passphrase for /home/USER/.ssh/id_rsa:
```

После выхода из системы парольная фраза будет забыта. Команду необходимо вводить каждый раз при подключении к виртуальной консоли или окну терминала.



Для подключения к серверу OpenSSH с клиентской машины, необходимо установить пакет **openssh-clients**.

Использование утилиты ssh.

Утилита ssh позволяет входить на удаленную машину и выполнять на ней команды. Это безопасная замена для таких программ как **rlogin**, **rsh** и **telnet**.

Подобно команде **telnet**, для входа на удаленную машину используется команда следующего вида:

```
$ ssh hostname
```

Например, для входа на удаленную машину **server01.ll-100.local**, введите следующую команду:

```
$ ssh server01.ll-100.local
```

При таком подключении вы используете для входа то же имя пользователя, под которым работаете на локально системе. Если необходимо указать другое имя, используется команда следующего вида:

```
$ ssh USERNAME@HOSTNAME
```

Например, для входа на удаленную машину **server01.ll-100.local** от имени пользователя **student** введите следующую команду:

```
$ ssh student@server01.ll-100.local
```

При первой инициализации подключения, выдается сообщение, подобное следующему:

```
The authenticity of host 'penguin.example.com' can't be established.
ECDSA key fingerprint is 256
da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff.
```

```
Are you sure you want to continue connecting (yes/no)?
```

Пользователи должны проверить корректность цифрового отпечатка перед ответом на запрос. Пользователь может запросить у администратора сервера подтверждение корректности ключа. Если пользователь может получить прямой доступ к хосту, то для получения цифрового отпечатка, можно воспользоваться командой ssh-keygen следующим образом:

```
# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff (ECDSA)
```

Введите yes для принятия ключа и подтверждения подключения. Вы увидите уведомление, что сервер был добавлен в список известных хостов и запрос на ввод пароля:

```
Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of
known hosts.
USER@ penguin.example.com's password:
```

ПРИМЕЧАНИЕ

Если ключ хоста сервера SSH изменился, клиент уведомляет пользователя, что подключение не может быть установлено до тех пор пока старый ключ хоста не будет удален из файла **~/.ssh/known_hosts**. Прежде чем сделать это, уточните у администратора сервера, не произошла ли подмена.

Для удаления ключа из файла **~/.ssh/known_hosts**, воспользуйтесь следующей командой:

```
$ ssh-keygen -R server01.ll-100.local
# Host c7-server01.ll-100.local found: line 15 type ECDSA
/home/student/.ssh/known_hosts updated.
Original contents retained as
/home/USER/.ssh/known_hosts.old
```

После ввода пароля, будет предоставленастрока оболочки удаленной машины.

В качестве альтернативы, программа **ssh** может выполнить команду на удаленной машине без входа в строку оболочки:

```
$ ssh [USERNAME@]HOSTNAME COMMAND
```

Например, файл **/etc/redhat-release** предоставляет информацию о версии RedOS. Для просмотра содержимого этого файла на удаленной машине **server01.ll-100.local**, воспользуйтесь следующей командой:

```
$ ssh student@server01.ll-100.local cat /etc/redhat-release
student@server01.ll-100.local's password:
RedOS 7.2
```

После ввода корректного пароля и вывода результатов выполнения команды на удаленной машине вы вернетесь в локальную строку оболочки.

Использование утилиты scp.

Утилита **scp** может быть использована для передачи файлов между машинами по безопасному, зашифрованному соединению. В общем виде она очень похожа на **rcp**.

Для передачи файла на удаленную систему, используйте команду в следующей форме:

```
$scp LOCALFILE USERNAME@HOSTNAME:REMOTEFILE
```

Например, если вы хотите передать **file.big** на удаленную машину с именем **server01.ll-100.local**, воспользуйтесь следующей командой:

```
# scp server01:/root/file.big /root/
The authenticity of host 'server01 (192.168.0.21)' can't be established.
ECDSA key fingerprint is ab:9a:7f:eb:25:3b:3d:f2:c8:63:b5:03:02:2f:0d:62.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server01' (ECDSA) to the list of known hosts.
root@c7-server01's password:
file.big                                      100%   43MB  43.1MB/s   00:01
```

Несколько файлов может быть указано за раз.

Для перемещения удаленного файла на локальную систему, используйте следующий синтаксис:

```
scp USERNAME@HOSTNAME:REMOTEFILE LOCALFILE
```

Использование утилиты sftp.

Утилита **sftp** может быть использована для открытия безопасной, интерактивной сессии FTP. По своей структуре, она похожа на **ftp** за исключением того, что использует безопасное, зашифрованное подключение.

Для подключения к удаленной системе используйте команду в следующей форме:

```
$ sftp USERNAME@HOSTNAME
```

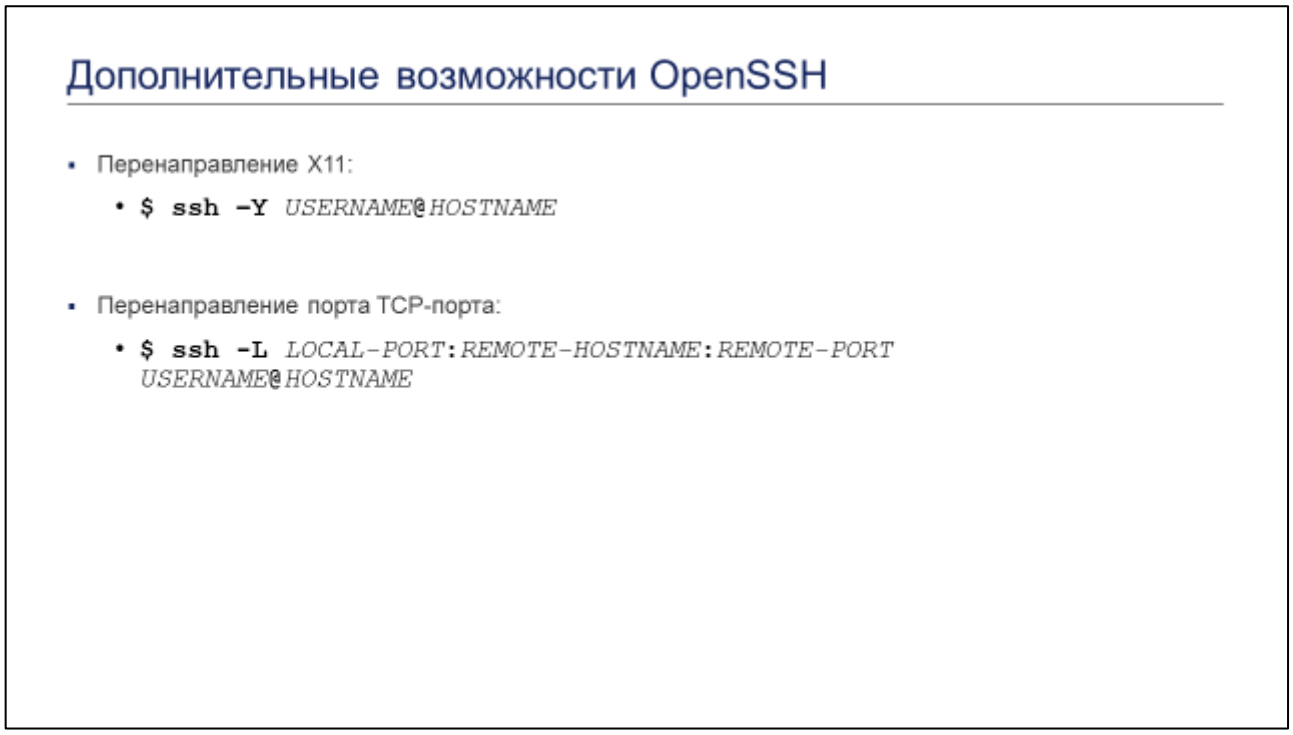
Например, для входа на удаленную машину **server01.ll-100.local** от имени учетной записи **student**, введите:

```
# sftp student@server01
student@server01's password:
Connected to server01.
sftp>
```

После ввода корректного пароля, будет представлена командная строка. Утилита **sftp** принимает набор команд, похожий на используемый **ftp**, некоторые из них представлены в таблице ниже:

Команда	Описание
ls *[каталог]*	Вывод содержимого удаленного *каталога*, если каталог не указан, то по умолчанию используется текущий каталог.
cd *каталог*	Сменить удаленный рабочий каталог на *каталог*.
mkdir *каталог*	Создать удаленный каталог (*каталог*).
rmdir *путь*	Удалить удаленный каталог (*каталог*).
put *localfile [remotefile]*	Переместить локальный файл на удаленную машину.
get *[remotefile] localfile*	Переместить удаленный файл на локальную машину.

Для получения полного списка доступных команд, обратитесь к странице руководства **sftp(1)**.



Безопасный интерфейс командной строки – это только начало для множества возможностей использования SSH. При наличии достаточного объема пропускной способности, сессии X11 могут быть перенаправлены через канал SSH. Или, при помощи перенаправления TCP/IP, изначально небезопасные соединения между системами могут быть привязаны к указанным каналам SSH.

Перенаправление X11.

Для открытия сессии X11 через SSH подключение, воспользуйтесь командой в следующей форме:

```
$ ssh -Y username@hostname
```

Например, для входа на удаленную машину (**client02.ll-100.local**) от имени пользователя student, введите:

```
$ ssh -Y student@client02.ll-100.local
```

Когда программа X запускается из строки безопасной оболочки (SSH), клиент SSH и сервер создают новый безопасный канал и данные программы X прозрачно пересылаются через этот канал.

ПРИМЕЧАНИЕ

Обратите внимание, что X Windows System должна быть установлена на удаленной системе до перенаправления X11. Введите следующую команду от имени пользователя root для установки группы пакетов X11:

```
# yum group install "X Window System"
```

Перенаправление X11 может быть очень удобным. Например, перенаправление X11 может быть использовано для создания безопасной интерактивной сессии с утилитой Print Settings. Чтобы сделать это, подключитесь к серверу при помощи ssh и введите:

```
$ system-config-printer &
```

Утилита Print Settings отобразится, позволив безопасно настроить печать на удаленной системе.

Перенаправление порта.

SSH может обеспечить безопасность прочих небезопасных протоколов TCP/IP через перенаправление портов. При использовании этой техники, сервер SSH выступает в роли зашифрованного тоннеля для клиента SSH.

Перенаправление порта работает за счет привязки локального порта клиента к удаленному порту на сервере. SSH может привязывать любой порт с сервера к любому порту на клиенте. Номера портов не обязаны соответствовать, чтобы эта техника работала.

ПРИМЕЧАНИЕ

Настройка перенаправления портов на прослушивание портов меньше 1024, требует уровня доступа root.

Чтобы создать канал перенаправления порта TCP/IP на локальной системе (localhost), используйте команду в следующей форме:

```
$ ssh -L LOCAL-PORT:REMOTE-HOSTNAME:REMOTE-PORT USERNAME@HOSTNAME
```

Например, для проверки поты на сервере **server01.ll-100.local**, при помощи POP3 через зашифрованное подключение, используйте следующую команду:

```
$ ssh –L 1100:server01.ll-100.local:110 server01.ll-100.local
```

После установки канала перенаправления порта между клиентской машиной и почтовым сервером, направьте почтового клиента POP3 на использование порта 1100 на **localhost** для проверки новых почтовых сообщений. Любые запросы, отправленные на порт 1100 на клиентской системе, будут безопасно направлены на сервер **server01.ll-100.local**.

Если на сервере **server01.ll-100.local** не запущен сервер SSH, но на других машинах в той же сети запущен, SSH может быть использован для обеспечения безопасности части подключения. Однако, потребуется частично изменить команду:

```
$ ssh –L 1100:server01.ll-100.local:110 server02.ll-100.local
```

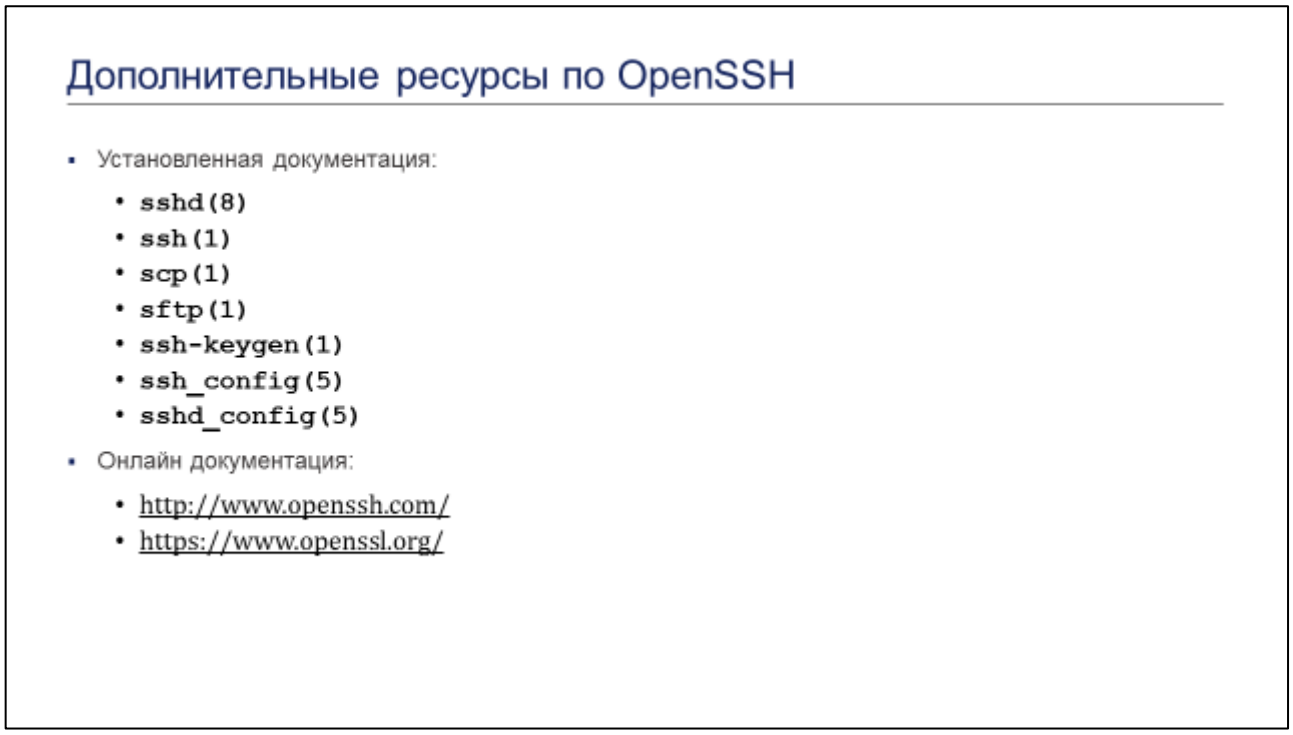
В примере, POP3 запрашиваемый на порту 1100 на клиентской машине, перенаправляется на порт 22 на сервере SSH (**server02.ll-100.local**). Затем, **server02.lll-00.local** соединяется с портом 110 на **server01.ll-100.local** для проверки нового почтового сообщения. Обратите внимание что при таком подключении безопасное соединение существует только между клиентской системой и **server02.lll-00.local**.

Перенаправление порта может также быть использовано для безопасного получения информации через сетевые экраны. Если сетевой экран настроен для пропуска трафика SSH через стандартный порт (22), но блокирует доступы к прочим портам, подключение между двумя хостами возможно за счет перенаправления коммуникаций через установленное SSH подключение.

ПРИМЕЧАНИЕ

Использование перенаправления порта для перенаправления подключения в конечном счете позволяет любому пользователю на клиентской системе подключаться к службе. Если клиентская система будет компрометирована, злоумышленник получит доступ к перенаправленным службам.

Системные администраторы знакомые с перенаправлением портов могут отключить эту возможность на сервере, указав **`No`** для параметра **`AllowTcpForwarding`** в конфигурационном файле **`/etc/ssh/sshd_config`** и перезапустив службу **`sshd`**.



Для получения дополнительной информации по настройке или подключению к серверу OpenSSH на системе RedOS, обратитесь к ресурсам, перечисленным ниже.

Установленная документация.

- **sshd(8)** – Страница руководства демона **sshd**, содержит доступные опции командной строки и предоставляет полный список поддерживаемых конфигурационных файлов и каталогов.
- **ssh(1)** – Страница руководства клиентского приложения **ssh**, предоставляет полный список доступных опций командной строки, поддерживаемых конфигурационных файлов и каталогов.
- **scp(1)** – Страница руководства утилиты **scp**, предоставляет более детальное описание утилиты и ее использования.
- **sftp(1)** – Страница руководства утилиты sftp.
- **ssh-keygen(1)** – Страница руководства утилиты **ssh-keygen**, детально описывает ее использование для генерации, управления и конвертации ключей аутентификации, используемых **ssh**.
- **ssh_config(5)** – Страница руководства **ssh_config**, описывает доступные опции настройки клиента SSH.
- **sshd_config(5)** – Страница руководства **sshd_config**, предоставляет полное описание доступных опций настройки демона SSH.

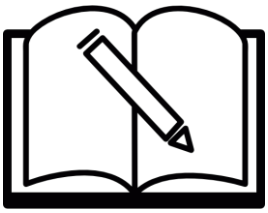
Онлайн документация.

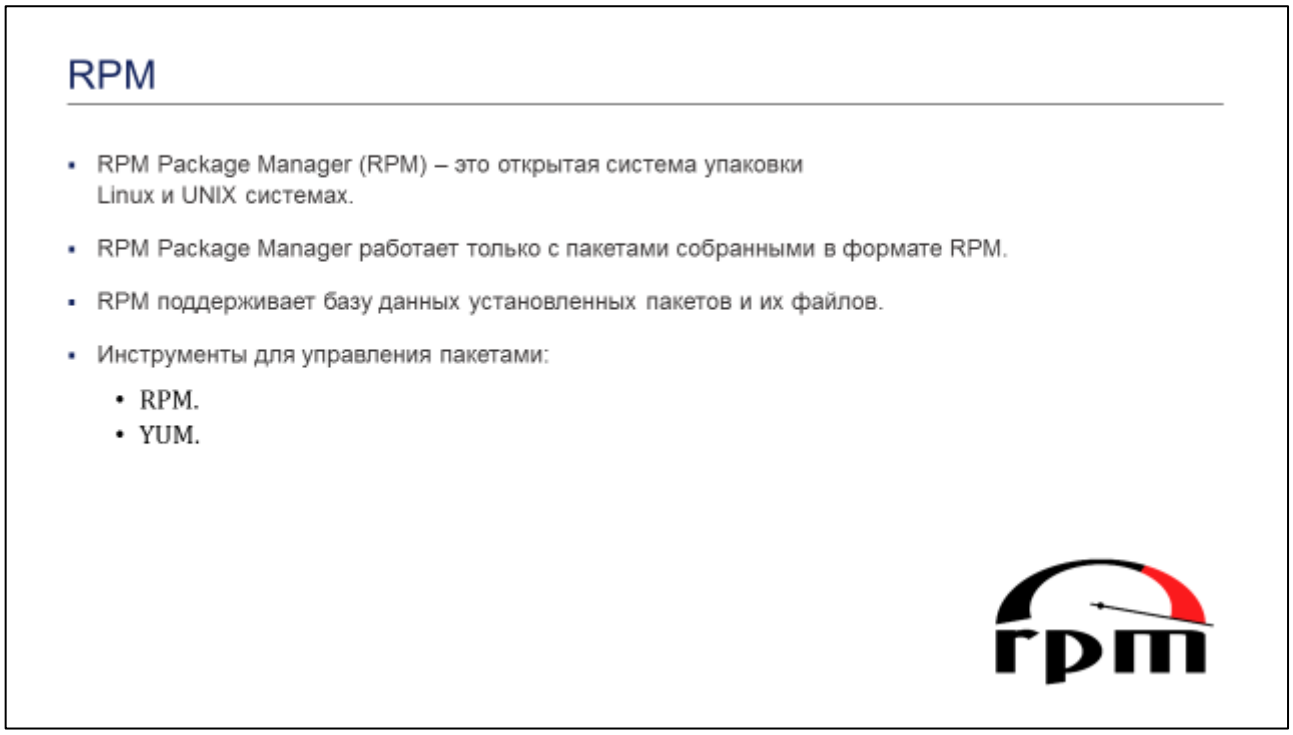
- http://www.openssh.com/ – Домашняя страница OpenSSH, содержит подробную документацию, часто задаваемые вопросы, ссылки на списки рассылок, отчеты об ошибках и прочие полезные ресурсы.
- https://www.openssl.org/ – Домашняя страница OpenSSL, содержит подробную документацию, часто задаваемые вопросы, ссылки на списки рассылок, отчеты об ошибках и прочие полезные ресурсы.

Модуль 7. Управление программным обеспечением РЕД ОС

В этом модуле рассматриваются:

- Управление пакетами RPM
- Управление репозиториями YUM
- Сборка программ из исходных кодов





RPM Package Manager (RPM) – это открытая система упаковки, которая используется в RedOS и других Linux и UNIX системах. RPM распространяется под лицензией GPL (GNU General Public License).

RPM Package Manager работает только с пакетами, собранными в формате RPM. RPM предоставляется как предустановленный пакет. Для конечных пользователей RPM упрощает обновление системы. Установка, удаление и обновление пакетов RPM может быть выполнена при помощи короткой команды. RPM поддерживает базу данных установленных пакетов и их файлов, что позволяет выполнять запросы и проверять систему. Также в системе присутствует ряд приложений, таких как Yum или PackageKit, которые упрощают работу с пакетами в формате RPM.

ПРЕДУПРЕЖДЕНИЕ

Для большинства задач управления пакетами, пакетный менеджер Yum предлагает эквивалент, как правило с большими возможностями. Yum также отслеживает и разрешает зависимости пакетов. Yum поддерживает целостность системы, принудительно проверяя целостность при установке или удалении пакетов при помощи сторонних приложений, таких как RPM. Настоятельно рекомендуется использовать Yum для задач управления пакетами вместо RPM.

ПРИМЕЧАНИЕ

При установке пакета, убедитесь, что он совместим с вашей архитектурой процессора. Обычно это легко определить при помощи имени пакета. Например, имя файла пакета, скомпилированного под архитектуру AMD64/Intel 64 заканчивается на **`x86_64.rpm`**.

В процессе обновления пакета, RPM осторожно обрабатывает конфигурационные файлы таким образом, чтобы вы не потеряли ваши изменения – то, чего не возможно добиться с обычными файлами **`.tar.gz`**.

Для разработчиков, RPM позволяет взять исходные коды и упаковать их в исходные и бинарные пакеты для конечных пользователей. Этот процесс существенно упрощает доставку приложений и предоставляет единый файл и опциональные обновления. Возможность упаковки исходных файлов и инструкций по обновлению упрощает поддержку пакетов и обновление до последних версий.

ПРИМЕЧАНИЕ

Так как RPM может вносить изменения в систему, выполняя установку, обновление, понижение версий и установку бинарных системных пакетов, в большинстве случаев требуются привилегии root. В настройках по умолчанию исключения составляют только домашний каталог и каталоги, созданные администратором (с соответствующими правами).

Для того чтобы понять, как использовать RPM, будет полезно понять цели проекта RPM.

Возможность обновления.

При помощи RPM можно обновить отдельный компонент системы без полной переустановки системы. После релиза новой версии операционной системы RPM позволяет интеллектуально, полностью автоматически обновить систему на месте (In-place Upgrade). Дополнительно, конфигурационные файлы будут сохранены неизменными в процессе обновления, таким образом, чтобы вы не потеряли ваши настройки. Для обновления не требуются дополнительные файлы, так как и обновление и установка производится из одних и тех же файлов.

Мощные запросы.

RPM спроектирован для предоставления мощной системы создания запросов. Присутствует возможность создания запросов из базы данных по установленным пакетам и или их файлам. Вы можете легко определить из какого пакета взялся тот или иной файл, а также откуда взялся тот или иной пакет. Файлы, содержащиеся в RPM-пакете, хранятся в сжатом архиве с бинарным заголовком, содержащем полезную информацию о пакете и его содержимом, что позволяет легко и быстро делать запросы к отдельному пакету.

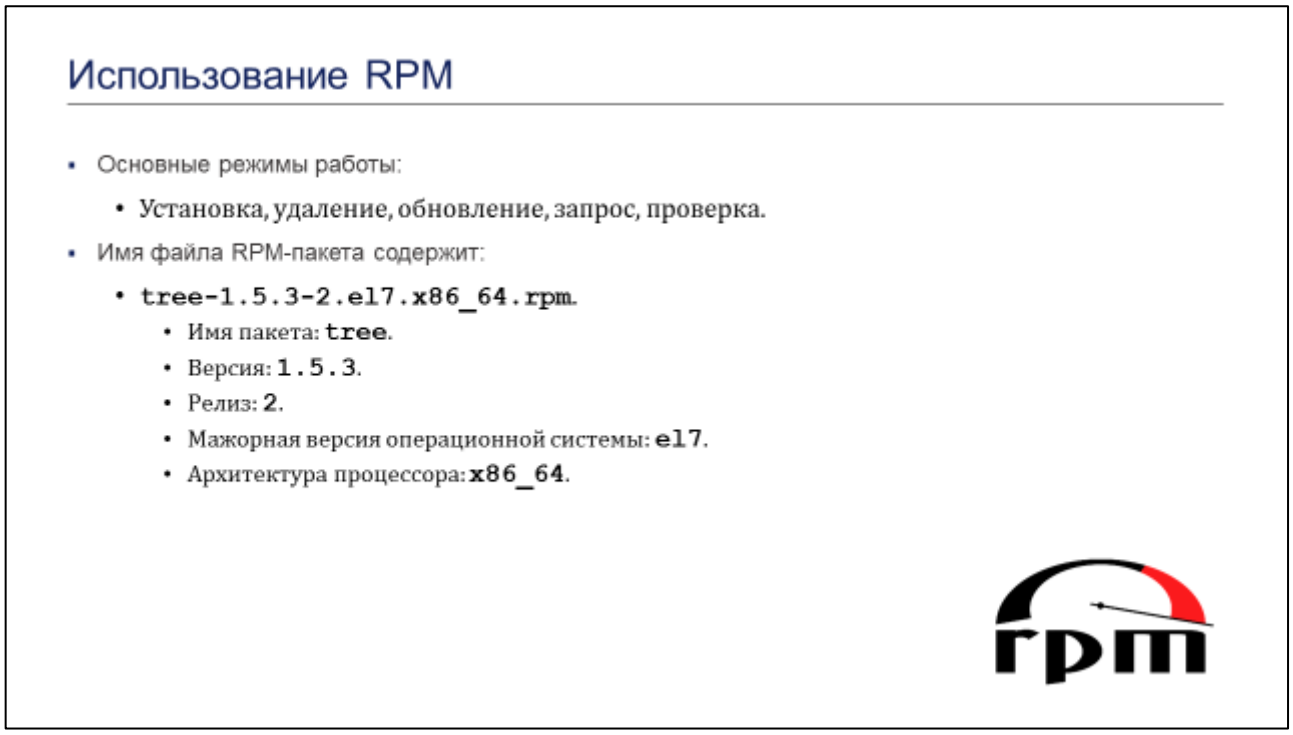
Системная проверка.

Другая мощная возможность RPM – это возможность проверять пакеты. Если случайно был удален важный файл, какого-либо установленного пакета, вы можете проверить пакет. Вы будете уведомлены обо всех аномалиях и сможете переустановить пакет. Любые конфигурационные файлы, которые вы изменяли, будут сохранены в процессе переустановки.

Оригинальные исходники.

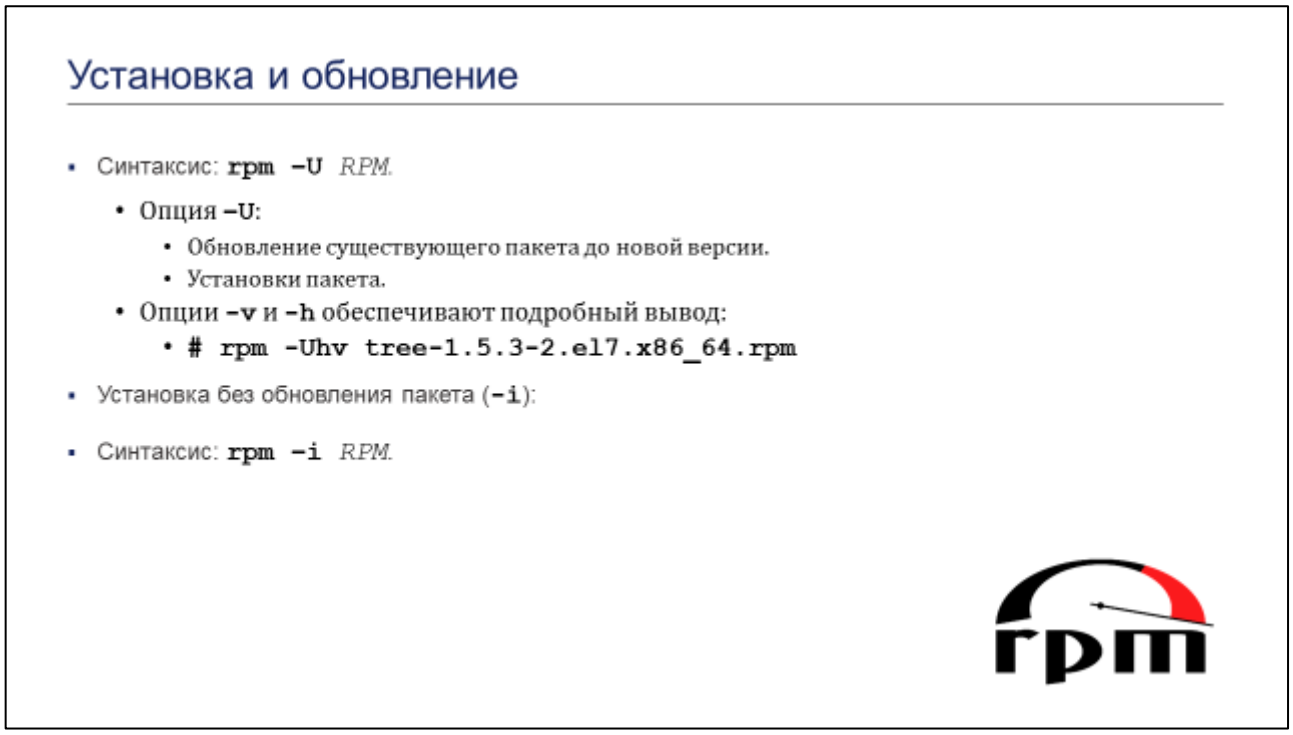
Критически важная цель в проектировании RPM – обеспечение возможности использования нетронутых исходных кодов программного обеспечения, такими, какими их заложили авторы. Вместе с RPM у вас имеются все исходные коды, распространяемые вместе со всеми обновлениями,

которые могут быть использованы с инструкциями по сборке. Это важное преимущество, которое может быть использовано для различных целей. Может показаться, что данное преимущество имеет значение только для разработчиков, но на самом деле это гарантирует высокое качество программного обеспечения.



RPM имеет пять основных режимов работы (за исключением сборки пакета): установка, удаление, обновление, запрос и проверка. Далее мы рассмотрим каждый из этих режимов. Для получения полной информации обо всех опциях используйте **`rpm --help`** или **`man rpm`**.

RPM-пакеты, как правило имеют имя похожее на **`tree-1.5.3-2.el7.x86_64.rpm`**. Имя файла содержит имя пакета (**`tree`**), версию (**`1.5.3`**), релиз (**`2`**), мажорную версию операционной системы (**`el7`**) и архитектуру CPU (**`x86_64`**).



Вы можете использовать опцию **–U** для:

- Обновления существующего, старого пакета до новой версии.
- Установки пакета.

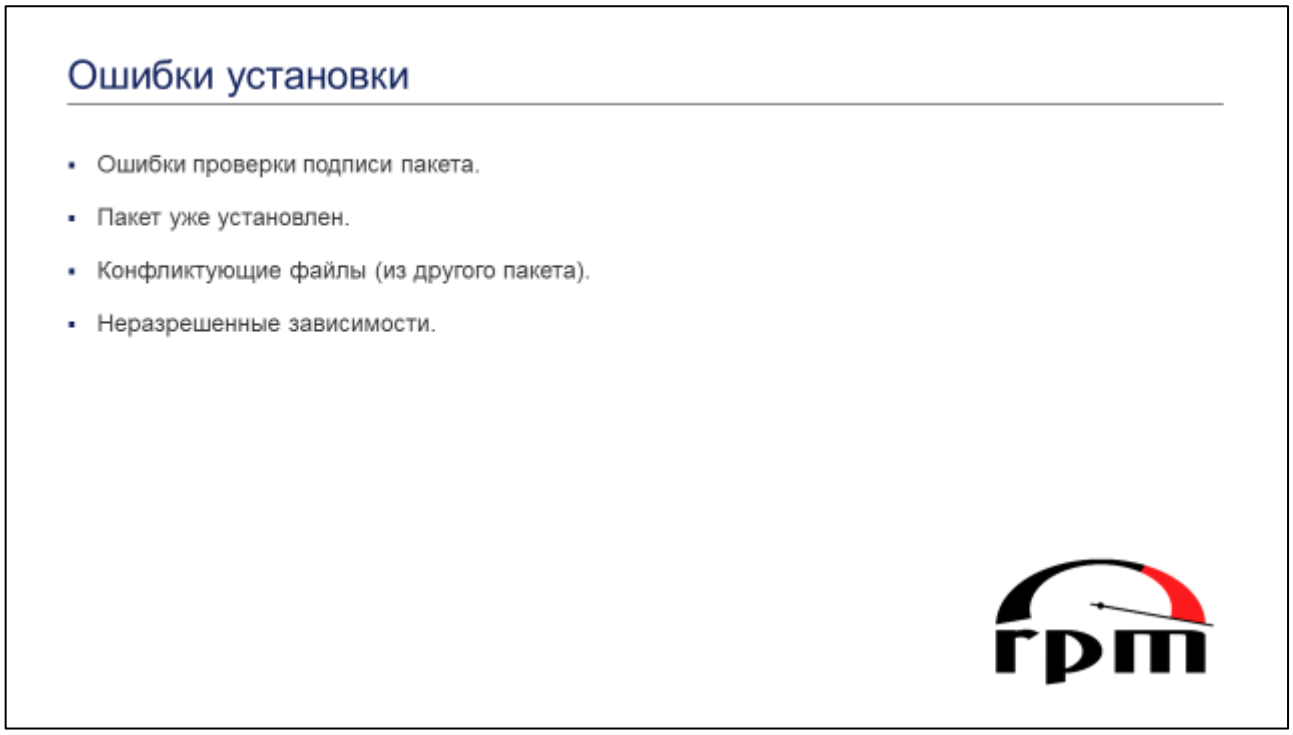
Синтаксис: **rpm –U** *RPM_FILE*

Для установки или обновления пакета **tree-1.5.3-2.el7.x86_64.rpm**, выполните следующую команду с правами пользователя root:

```
# rpm -Uhv tree-1.5.3-2.el7.x86_64.rpm
```

ПРИМЕЧАНИЕ

Опции **-v** и **-h** дают указание rpm вывести более подробный вывод, отображая прогресс при помощи **#**.



Если обновление/установка прошли успешно, следующий вывод будет отображен.

```
# rpm -Uhv tree-1.5.3-2.el7.x86_64.rpm
Preparing...  ########################################### [100%]
1:tree        ########################################### [100%]
```

ПРЕДУПРЕЖДЕНИЕ

rpm предоставляет две разные опции для установки пакетов: опцию **-U** (исторически обозначающую update) и опцию **-i**, предназначенную для установки. Настоятельно рекомендуется использовать опцию **-U** для установки всех пакетов за исключением ядра операционной системы Linux.

Всегда используйте опцию **-i** (install) для установки пакета нового ядра, вместо обновления. Так как опция **-U** обновляет ядро, удалив старое ядро и, если что-то пойдет не так вы не сможете загрузиться с использованием старого ядра. Поэтому всегда используйте команду **rpm -i** *ПАКЕТ_ЯДРА*.

Проверка подписи пакета.

Подпись пакета проверяется автоматически при установке и обновлении пакета. Подпись подтверждает, что пакет подписан авторизованной стороной. Например, если проверка подписи завершится неудачно, вы получите следующее сообщение:

```
error: tree-1.5.3-2.el7.x86_64.rpm : Header V3 RSA/SHA256 signature: BAD, key ID
d22e77f2
```

Если на системе не установлен необходимый ключ для проверки подписи, вы получите следующее сообщение, содержащее слово **NOKEY**:

```
warning: tree-1.5.3-2.el7.x86_64.rpm : Header V3 RSA/SHA1 signature: NOKEY, key ID
57bbccba
```

Пакет уже установлен.

Если пакет с таким же именем и такой же версией уже установлен вы получите следующий вывод:

```
Preparing... ########################################### [100%]
package tree-1.5.3-2.el7.x86_64 is already installed
```

Однако, если вы хотите переустановить пакет, используйте опцию **--replacepkgs**, которая заставляет RPM игнорировать ошибки.

```
# rpm -Uhv --replacepkgs tree-1.5.3-2.el7.x86_64.rpm
```

Эта опция полезна, если файлы из пакета были удалены или вы хотите вернуть оригинальные конфигурационные файлы из пакета.

Конфликтующие файлы.

Если вы попытаетесь установить пакет, который содержит файлы, которые уже были установлены из другого пакета, вы получите следующее сообщение:

```
Preparing... ################################################
file /usr/bin/foobar from install of foo-1.0-1.el7.x86_64 conflicts
with file from package bar-3.1.1.el7.x86_64
```

Для того, чтобы проигнорировать такую ошибку, воспользуйтесь опцией --replacefiles:

```
# rpm -Uvh --replacefiles foo-1.0-1.el7.x86_64.rpm
```

Неразрешенные зависимости.

Иногда RPM-пакет может зависеть от других пакетов, это значит, что другие пакеты должны быть установлены на системе, чтобы приложение запускалось корректно. Если вы попробуете установить пакет, который содержит неразрешенные зависимости, вы получите следующий вывод:

```
error: Failed dependencies:
bar.so.3()(64bit) is needed by foo-1.0-1.el7.x86_64
```

Если вы производите установку с установочного накопителя вы можете найти зависимые пакеты на нем или выгрузить их из иных источников. Для того чтобы установить пакет вместе с зависимостью, добавить название зависимого пакета в команду **rpm**:

```
# rpm -Uvh foo-1.0-1.el7.x86_64.rpm bar-3.1.1.el7.x86_64.rpm
```

Если установка обоих пакетов завершится успешно, вы получите следующий вывод:

```
Preparing... ########################################### [100%]
1:foo        ########################################### [ 50%]
2:bar        ########################################### [100%]
```

Вы можете использовать опцию **--whatprovides** для определения пакета, содержащего необходимый файл:

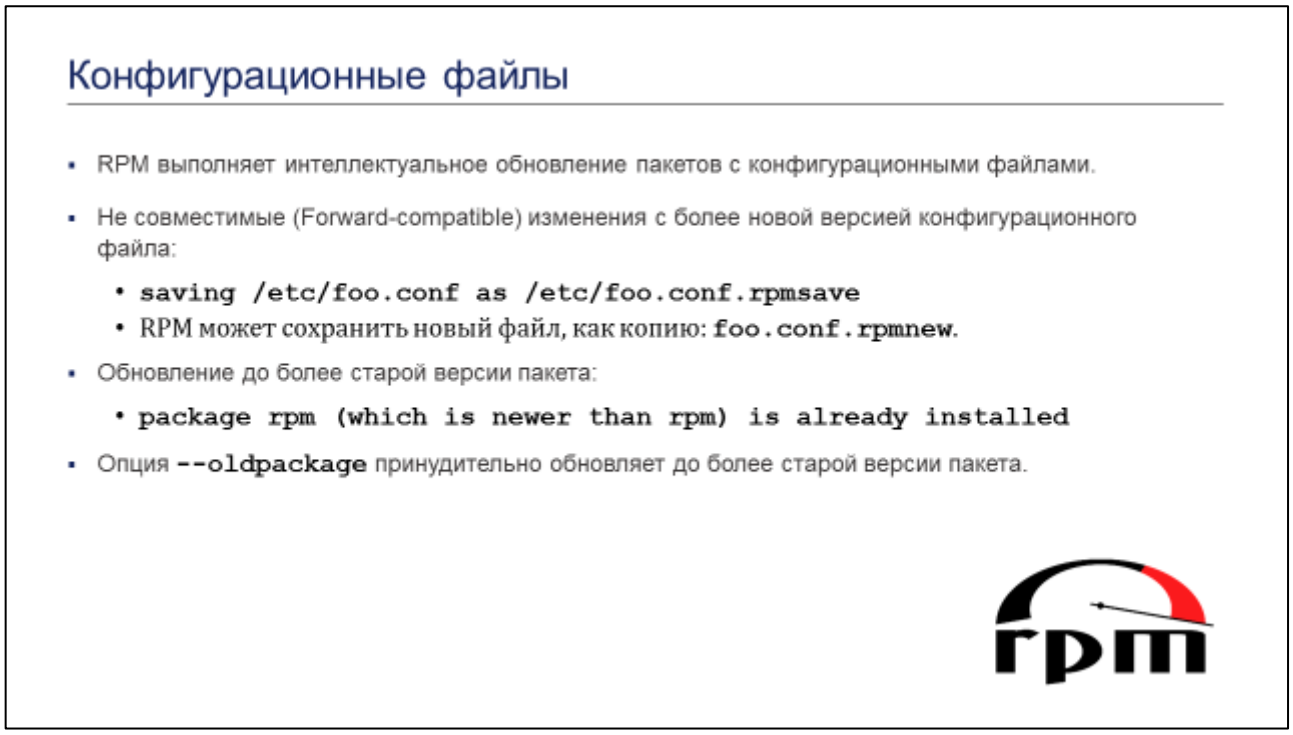
```
# rpm -q --whatprovides "bar.so.3"
```

Если пакет, который содержит файл bar.so.3 содержится в базе данных RPM, имя пакета будет отображено:

```
bar-3.1.1.el7.i586.rpm
```

ПРЕДУПРЕЖДЕНИЕ

Также вы можете принудительно поставить пакет, который содержит неразрешенные зависимости (при помощи опции **--node**), но этого не рекомендуется делать, так как обычно такая установка не позволяет запускаться приложению. Установка или удаление пакетов с использованием опции **--node** может привести к неисправности приложений и серьезным проблемам с управлением пакетов, в том числе к системным ошибкам.



Так как RPM выполняет интеллектуальное обновление пакетов с конфигурационными файлами, вы можете увидеть одно или несколько подобных сообщений:

```
saving /etc/foo.conf as /etc/foo.conf.rpmsave
```

Данное сообщение обозначает, что изменения, которые вы внесли в конфигурационный файл, не совместимы (Forward-compatible) с более новой версией конфигурационного файла в пакете, RPM в свою очередь сохранил измененный конфигурационный файл и установил новый. Далее необходимо вручную исправить новый конфигурационный файл, чтобы система оставалась настроенной.

Альтернативно RPM может сохранить новый конфигурационный файл, как копию, например, **`foo.conf.rpmnew`**, и оставить измененный конфигурационный файл нетронутым. Далее необходимо будет разрешить конфликты конфигурационного файла, например, совместив различия при помощи программы diff.

Если вы пробуете обновиться до более старой версии пакета (чем установленная в системе), вы получите следующее сообщение:

```
package foo-2.0-1.el7.x86_64.rpm (which is newer than foo-1.0-1) is already
installed
```

Для принудительного обновления до более старой версии пакета используйте опцию **`--oldpackage`**:

```
# rpm -Uvh --oldpackage foo-1.0-1.el7.x86_64.rpm
```

Удаление пакета происходит также просто, как и установка. При помощи следующей команды:

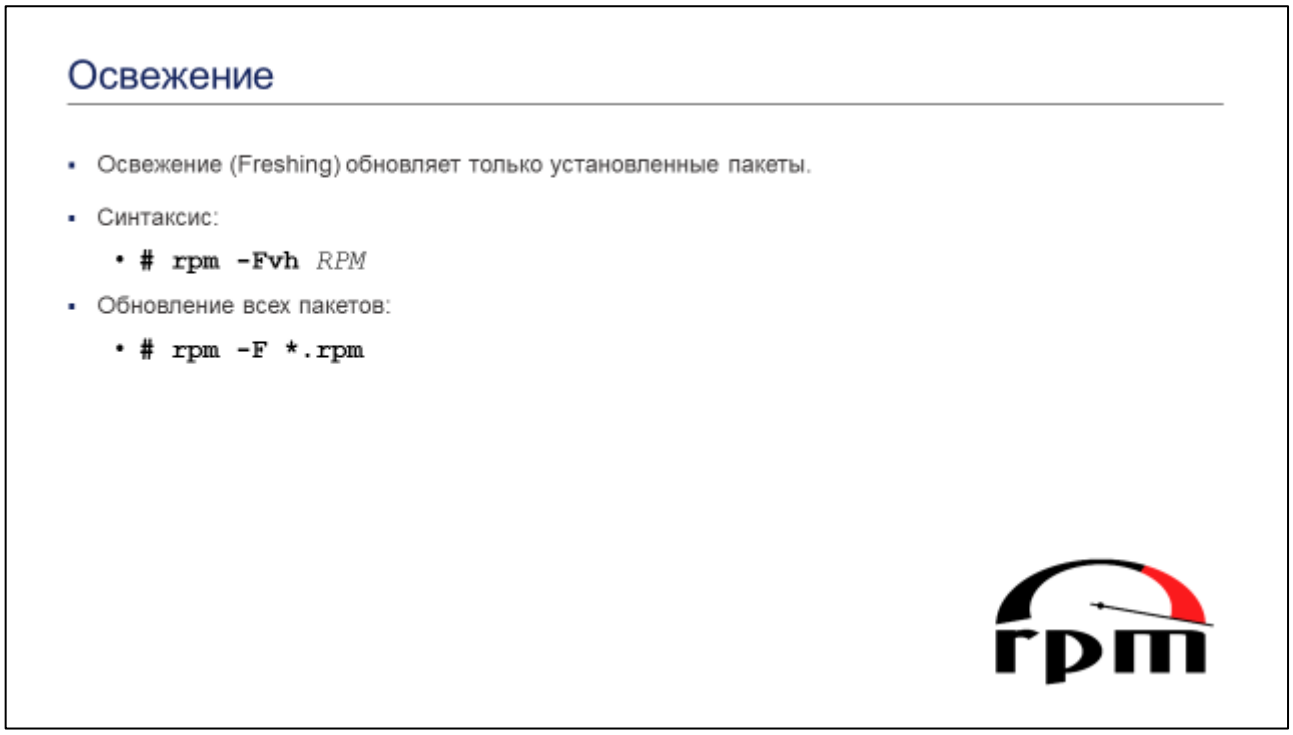
```
# rpm -e foo
```

ПРИМЕЧАНИЕ

Обратите внимание, что команда использует имя пакета **foo**, а не имя оригинального файла пакета **foo-1.0-1.el7.x86_64**. Если вы попытаетесь удалить пакет при помощи команды **rpm -e** и полного имени файла, то вы получите ошибку.

Вы можете столкнуться с ошибками зависимостей при удалении пакета, если от него зависят другие пакеты, например:

```
# rpm -e ghostscript
error: Failed dependencies:
libgs.so.8()(64bit) is needed by (installed) libspectre-0.2.2-3.el7.x86_64
libgs.so.8()(64bit) is needed by (installed) foom atic-4.0.3-1.el7.x86_64
libijs-0.35.so()(64bit) is needed by (installed) gutenprint-5.2.4-
5.el7.x86_64
ghostscript is needed by (installed) printer-filters-1.1-4.el7.noarch
```



Освежение (Freshing) похоже на обновление, за исключением того, что обновляются только установленные пакеты. Для освежения используйте следующую команду:

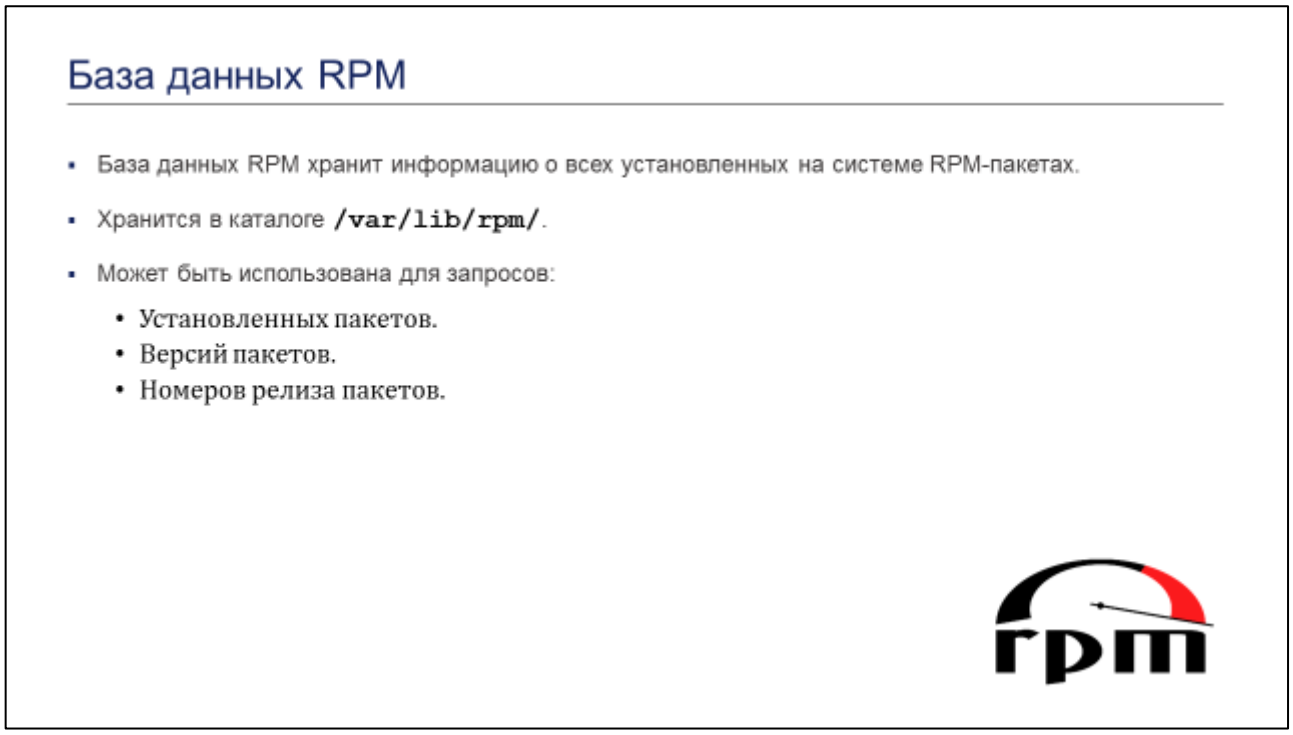
```
# rpm -Fvh foo-2.0-1.el7x86_64.rpm
```

Опция освежения пакета проверяет версию пакета, указанного в команде и сравнивает с версией установленного в системе пакета. Когда опция освежения обрабатывает пакет более новой версии, чем установленный в системе, пакет будет обновлен до новой версии. Однако, опция освежения не устанавливает пакет, если он отсутствует в системе, этим опция освежения (**-F**) отличается от опции обновления (**-U**).

В следующем примере обновляются все пакеты, для которых доступны обновления в текущей папке:

```
# rpm -F *.rpm
```

При этом обновляются только установленные в системе пакеты.



База данных RPM хранит информацию о всех установленных на системе RPM-пакетах. Она хранится в каталоге **/var/lib/rpm/**:

```
# ls /var/lib/rpm -1
Basenames
Conflictname
__db.001
__db.002
__db.003
Dirnames
Group
Installtid
Name
Obsoletename
Packages
Providename
Requirename
Sha1header
Sigmd5
Triggername
```

Каждый файл – это отдельная база данных в формате Berkeley DB, за исключением нескольких файлов данных **__db**.

ПРИМЕЧАНИЕ

Библиотеки Berkeley DB предоставляют простой API базы данных. Это не традиционная реляционная база данных. В ней значения данных хранятся в виде хеш-таблицы в парах имя/значение. Подобный тип баз данных очень быстр для просмотра именованных записей (таких как имена пакетов), но совсем не такой быстрый при итеративных запросах по всем записям.

Запросы

- Синтаксис: **rpm -q** *<имя_пакета>*
- Опции выбора пакета (Package Selection Options):
  - **-a** – запрос всех установленных в системе пакетов.
  - **-f** *<имя_файла>* - запрашивает информацию о принадлежности пакету файла *<имя_файла>*.
  - **-p** *<файл_пакета>* - запрашивает не установленный пакет *<файл_пакета>*.
- Опции запроса пакета (Package Query Options):
  - **-i** – отображают информацию о пакете.
  - **-l** – отображает список файлов, которые содержат пакет.
  - **-s** – отображает состояние всех файлов в пакете.
  - **-d** – отображает список файлов документации.
  - **-c** – отображает список конфигурационных файлов.
- Опция **-v** для отображения файлов в детальном формате.

К базе данных RPM можно обращаться, например, при помощи **rpm -q tree**, можно запросить информацию об установленном пакете tree и должны получить следующий вывод:

```
tree-1.5.2.2-4.el7.x86_64
```

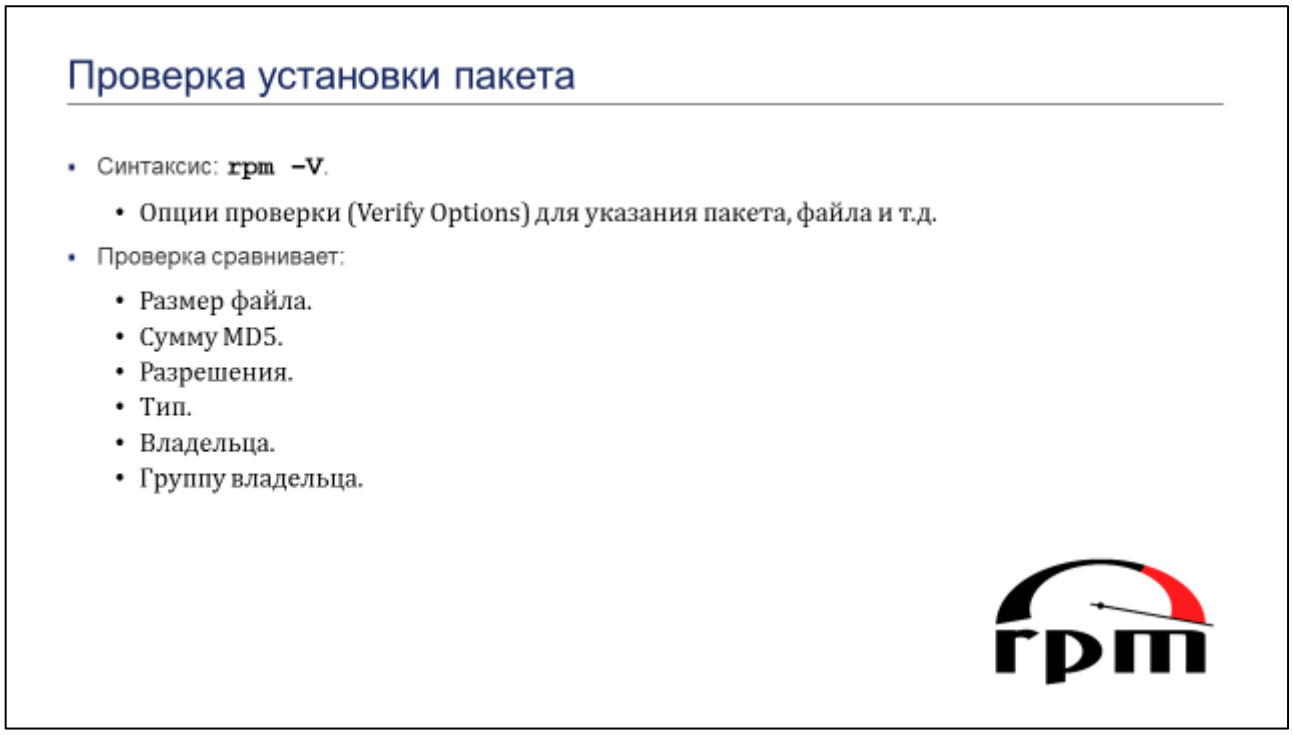
Вы можете также использовать следующие опции выбора пакета (Package Selection Options), которые описаны в страницах руководства man.

- **-a** – запрос всех установленных в системе пакетов.
- **-f** *ИМЯ_ФАЙЛА* - запрашивает из базы данных RPM информацию о принадлежности пакету файла *ИМЯ_ФАЙЛА* (Например, **rpm -qf /bin/ls**).
- **-p** *ФАЙЛ_ПАКЕТА* - запрашивает не установленный пакет *ФАЙЛ_ПАКЕТА*.

Есть несколько способов повлиять на отображаемую информацию о запрашиваемом пакете. Следующие опции используются для выбора типа информации для которой построен запрос. Они называются опции запроса пакета (Package Query Options).

- **-i** – Отображает информацию о пакете, в том числе имя, описание, релиз, размер, дату создания, дату установки и производителя.
- **-l** – Отображает список файлов, которые содержат пакет.
- **-s** – Отображает состояние всех файлов в пакете.
- **-d** – Отображает список файлов, помеченных как документация (страницы руководства man, страницы **info**, файлы **README** и т.д.).
- **-c** – отображает список файлов, помеченных как конфигурационные файлы. Это файлы, которые вы изменяете после установки для адаптации и настройки пакета на вашей системе (Пример: **sendmail.cf**, **passwd**, **initttab** и так далее).

Для опций, который отображают список файлов, добавьте опцию **-v**, чтобы отображать файлы в формате похожем на команду **ls -l**.



Проверка пакета, сравнивает информацию о файлах, установленных из пакета с информацией о файлах из оригинального пакета. Среди прочего, проверка сравнивает размер файла, сумму MD5, разрешения, тип, владельца и группу владельца каждого фала.

Команда **rpm -V** проверяет пакет. Вы можете использовать любую опцию проверки (Verify Options) из раздела про запросы, чтобы указать пакеты, которые необходимо проверить. Простое использование проверки – это: **rpm -V tree**, которое проверяет, что все файлы в пакете **tree** такие же, какими они были сразу после установки. Пример:

```
# rpm -Vf /usr/bin/tree
```

В данном примере **/usr/bin/tree** – это абсолютный путь до файла, который необходимо проверить.

Для проверки всех установленных на системе пакетов (потребуется некоторое количество времени), используйте следующую команду:

```
# rpm -Va
```

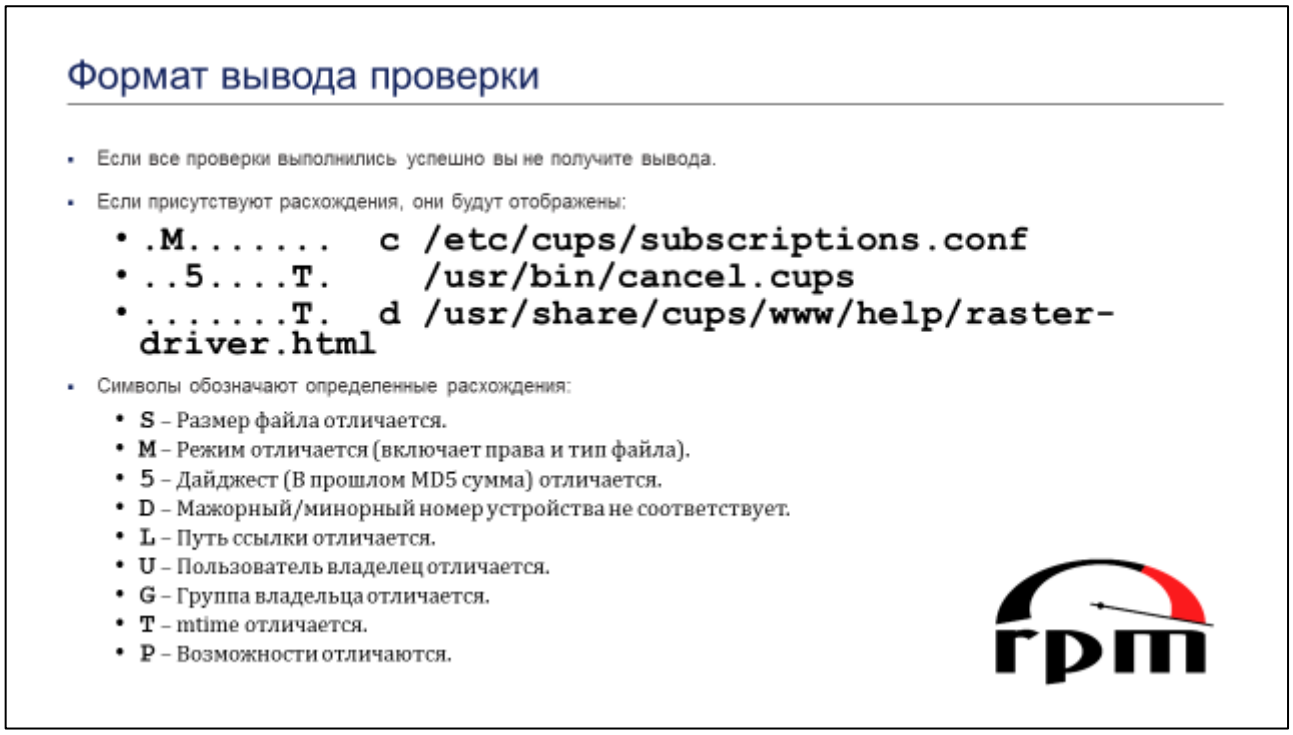
Для проверки установленного пакета при помощи файла пакета, используйте следующую команду:

```
# rpm -Vp tree-1.5.3-2.el7.x86_64.rpm
```

Данная команда может быть полезна, если вы подозреваете, что база данных RPM повреждена.

Если все проверки выполнились успешно вы не получите вывода. Если присутствуют расхождения, они будут отображены:

```
# rpm -Vp  cups-1.7.1-6.fc20.x86_64.rpm
.M.......  c /etc/cups/subscriptions.conf
..5....T.    /usr/bin/cancel.cups
.......T.  d /usr/share/cups/www/help/raster-driver.html
```



Формат вывода – это строка из девяти символов, идущая перед опциональным маркирующим атрибутом и именем файла.

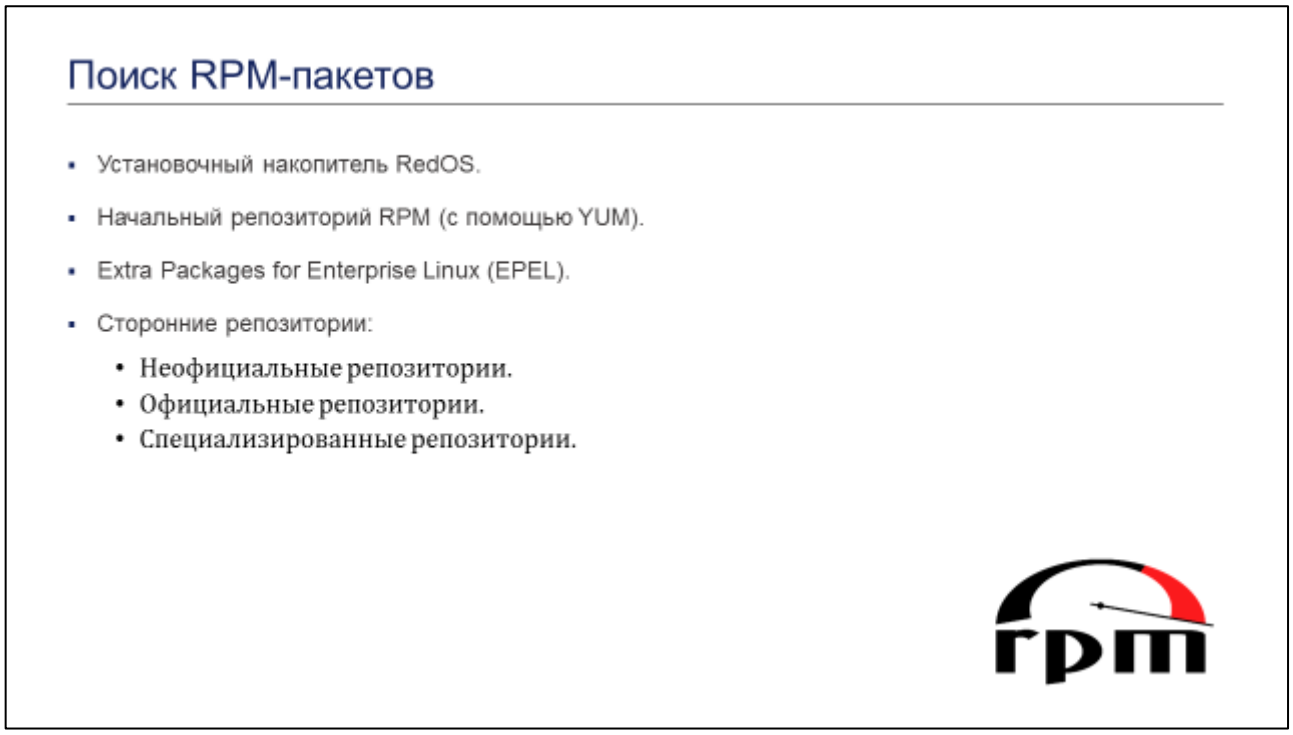
Первые 9 символов – это результат проверки, выполненной над файлом. Каждый тест – это сравнение одного атрибута файла со значением, записанным в базе данных RPM. Отдельно стоящая точка (`.`) обозначает, что тест пройден. Следующие символы обозначают определенные расхождения:

- **`S`** – Размер файла отличается.
- **`M`** – Режим отличается (включает права и тип файла).
- **`5`** – Дайджест отличается (в прошлом контрольная сумма MD5).
- **`D`** – Мажорный/минорный номер устройства отличается.
- **`L`** – Путь ссылки отличается.
- **`U`** – Пользователь владелец отличается.
- **`G`** – Группа владельца отличается.
- **`T`** – mtime отличается (время последнего изменения файла).
- **`P`** – Возможности отличаются.

Маркирующие атрибуты, если представлен, описывает назначение файла. Маркеры атрибутов включают в себя:

- **`c`** – Конфигурационный файл.
- **`d`** – Файл документации.
- **`l`** – Файл лицензии.
- **`r`** – Файл README.

В случае отображения любого вывода, используйте любой метод проверки установки пакета, если он удален, переустановите его для устранения проблемы.



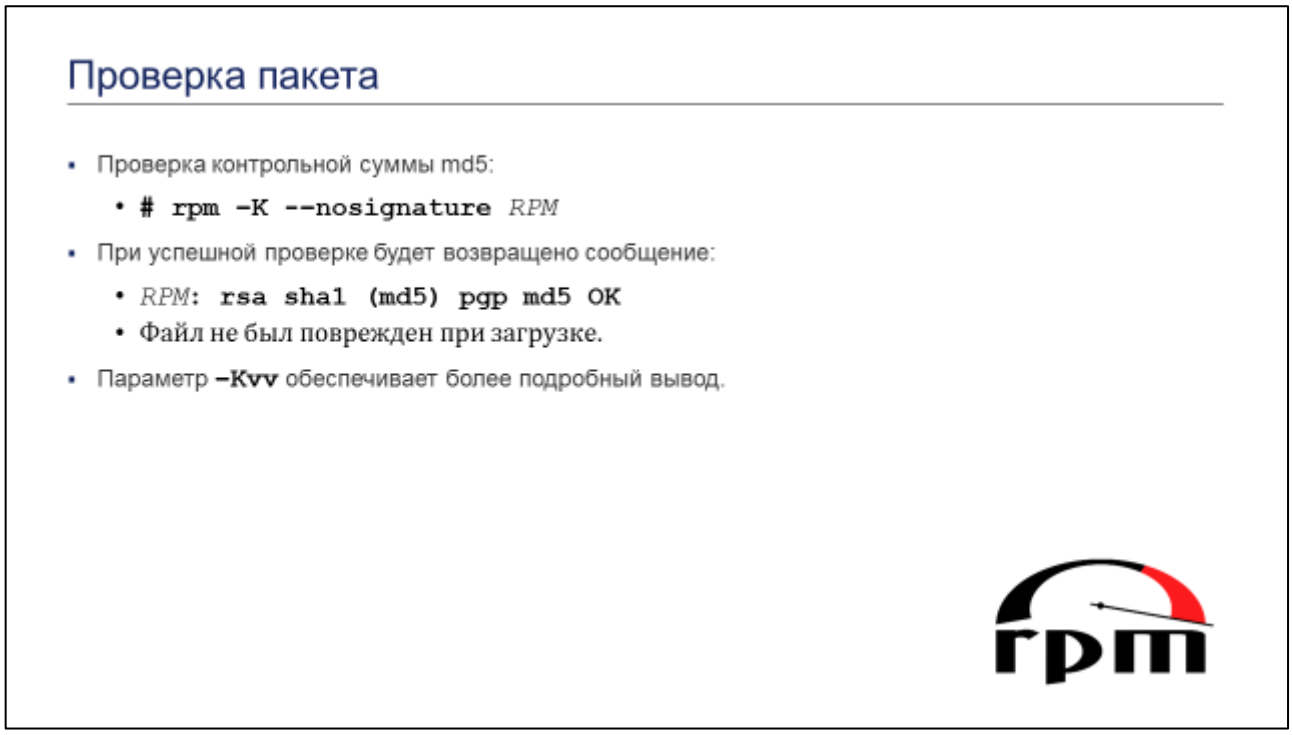
Прежде чем использовать какие-либо RPM-пакеты, необходимо научиться находить их и узнать, как доверять им.

Поиск в интернете возвращает множество репозиториев RPM, но если вы посмотрите на RPM-пакеты RedOS, они могут быть найдены в следующих расположениях:

- Установочный накопитель RedOS.
- Начальный репозиторий RPM предоставляемый с помощью пакетного менеджера YUM.
- Поддерживаемый сообществом, высококачественный источник дополнительных пакетов Extra Packages for Enterprise Linux (EPEL) Для получения дополнительной информации о EPEL, воспользуйтесь следующей ссылкой: http://fedoraproject.org/wiki/EPEL.
- Сторонние репозитории.
    - Неофициальные репозитории (например, REMI).
    - Официальные репозитории (например, RedHat и Oracle).
    - Специализированные репозитории (например, ZABBIX или MariaDB).

ПРИМЕЧАНИЕ

Когда вы думаете об использовании стороннего репозитория для вашей системы с RedOS, внимательно ознакомьтесь с веб-сайтом репозитория, обратив внимание на совместимость, прежде чем добавлять этот репозиторий в качестве источника пакетов. Репозиторий может содержать отличные версии пакетов от репозитория по умолчанию, что в конечном итоге может привести к проблемам совместимости на вашей системе.

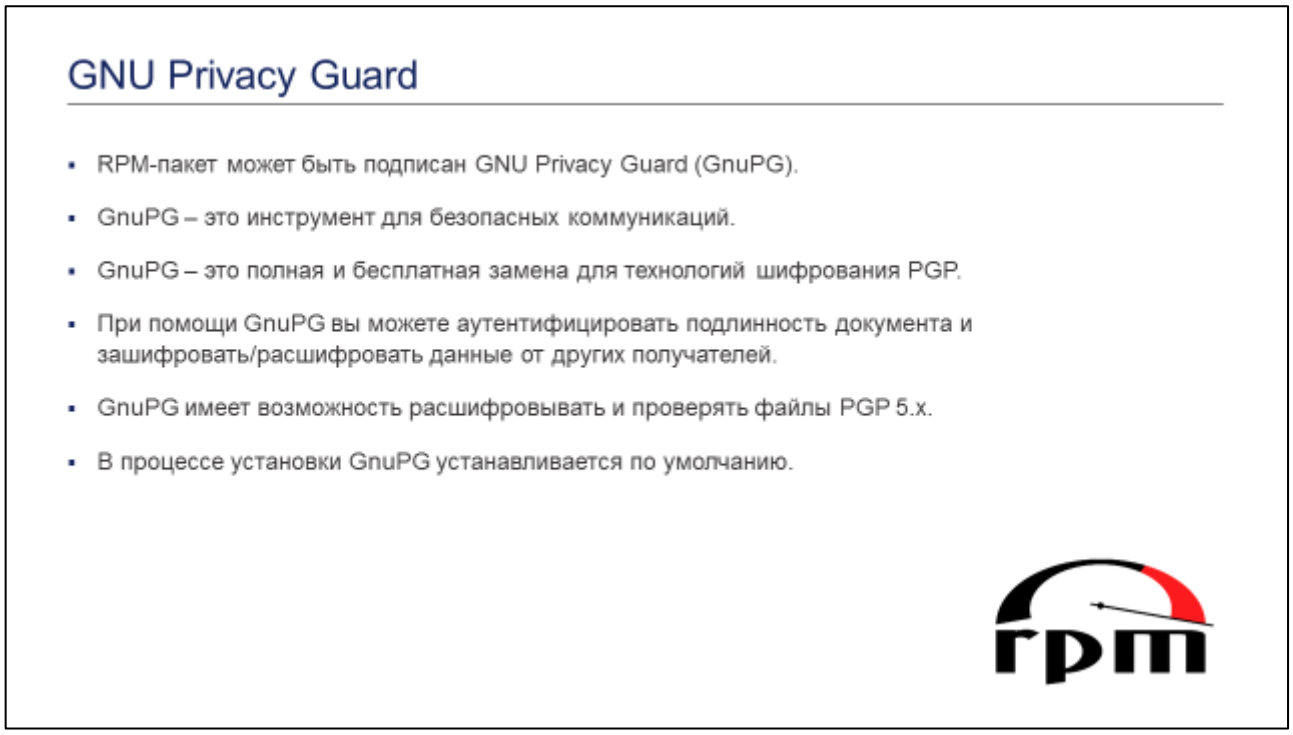


Если вы хотите убедиться, что пакет не был поврежден или специально модифицирован, проверьте контрольную сумму md5 при помощи следующей команды (где *RPM-ФАЙЛ* - это имя файла RPM-пакета):

```
# rpm -K --nosignature RPM-ФАЙЛ
```

Сообщение *RPM-ФАЙЛ*: **rsa sha1 (md5) pgp md5 OK** будет отображено. Это сообщение обозначает, что файл не был поврежден при загрузке. Для получения более подробного вывода замените параметр **-K** на **-Kvv**.

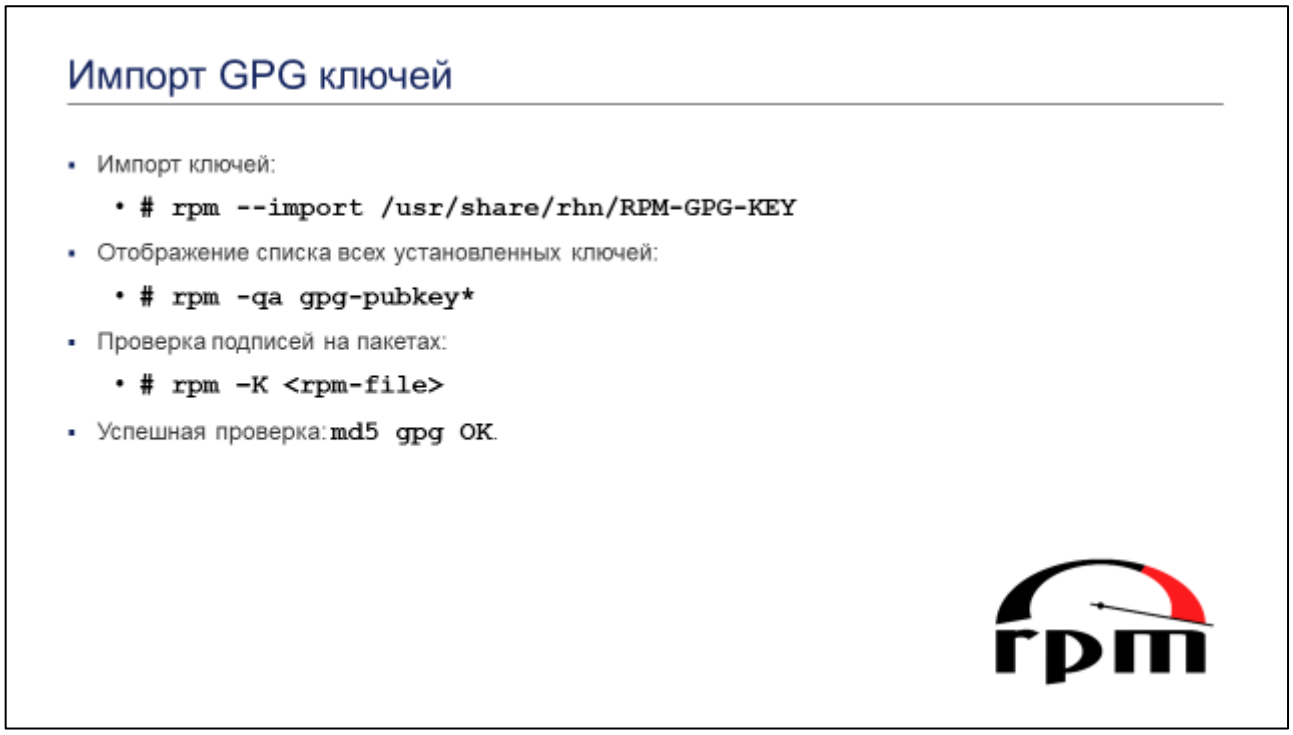
В остальных случаях возникает вопрос, как доверять разработчику, создавшему пакет? Если пакет был подписан ключом GnuPG разработчика, вы знаете, кем является разработчик на самом деле.



RPM-пакет может быть подписан при помощи GNU Privacy Guard (или GnuPG) для помощи в принятии решения о доверии скачанному пакету.

GnuPG – это инструмент для безопасных коммуникаций; это полная и бесплатная замена для технологий шифрования PGP. При помощи GnuPG вы можете аутентифицировать подлинность документа и зашифровать/расшифровать данные от других получателей. Также GnuPG имеет возможность расшифровывать и проверять файлы PGP 5.x.

В процессе установки GnuPG устанавливается по умолчанию. Поэтому вы можете немедленно приступить к использованию GnuPG для проверки любых пакетов, которые вы получили от RedOS. Прежде чем это делать, вы должны импортировать публичный ключ RedOS.



Для проверки пакетов, вы должны импортировать GnuPG ключ RedOS. Для этого выполните следующую команду:

```
# rpm --import /usr/share/rhn/RPM-GPG-KEY
```

Для отображения списка всех установленных ключей для проверки RPM, используйте следующую команду:

```
# rpm -qa gpg-pubkey*
```

Для ключа RedOS, вывод будет следующим:

```
# gpg-pubkey-db42a60e-37ea5438
```

Для отображения поддробностей об указанном ключе, используйте команду **`rpm -qi`** с указанием вывода предыдущей команды:

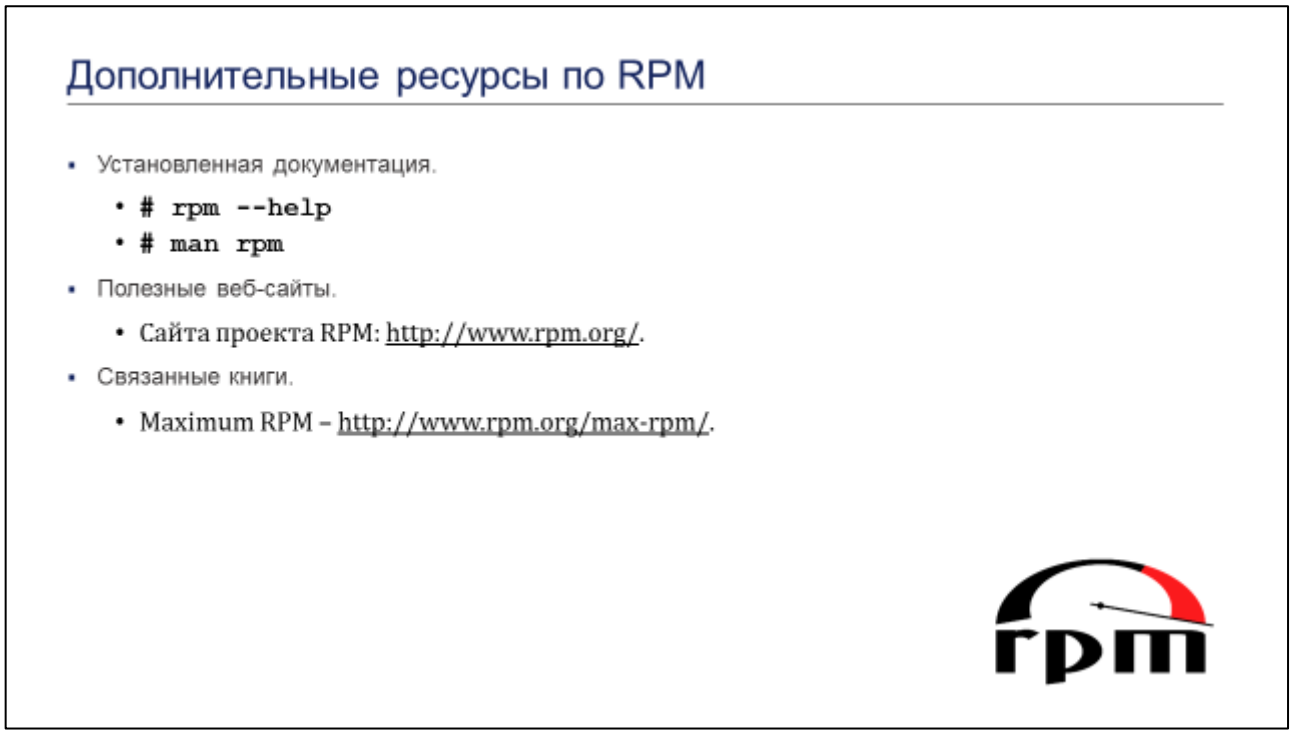
```
# rpm -qi gpg-pubkey-db4 2a60e-37ea54 38
```

Проверка подписей на пакетах.

Для проверки подписи GnuPG файла RPM после импорта ключа GnuPG создателя пакета, используйте следующую команду (Замените *`RPM-FILE`* на имя файла RPM-пакета):

```
# rpm -K RPM-FILE
```

Если все пройдет успешно, следующее сообщение будет отображено: **`md5 gpg OK`**. Это значит, что подпись пакета проверена, он не изменен и безопасен для установки и использования.



RPM это экстремально комплексная утилита с множеством опций и методов для запросов, установок, обновлений и удалений пакетов. Используйте следующие ресурсы для изучения возможностей RPM.

Установленная документация:

- **`rpm --help`** – отображает быструю подсказку по параметрам RPM.
- **`man rpm`** – отображает страницу руководства man, содержащую более детальную информацию о параметрах RPM.

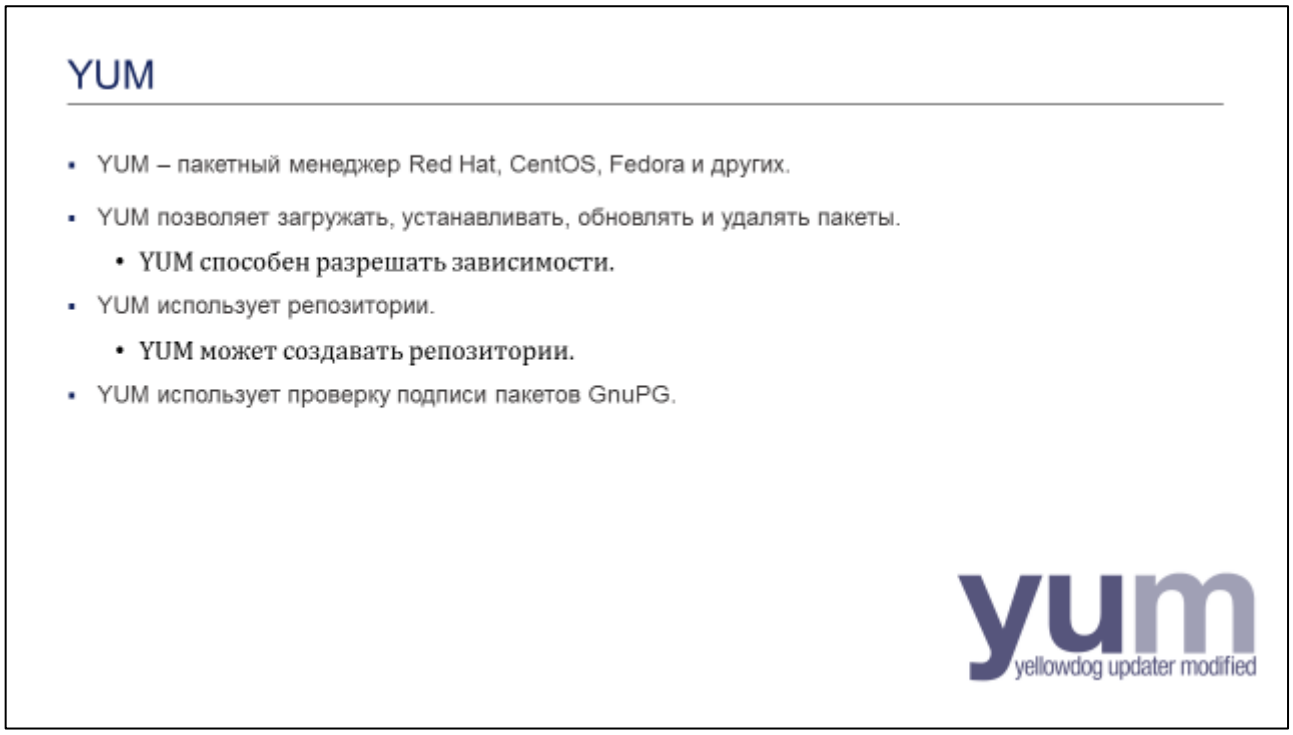
Полезные веб-сайты:

- Сайта проекта RPM: http://www.rpm.org/.

Связанные книги:

- Maximum RPM – http://www.rpm.org/max-rpm/.

Книга Maximum RPM, доступная для чтения в режиме онлайн, содержит полную информацию, начиная от использования RPM до создания своих собственных RPM-пакетов и программирования при помощи rpmlib.



Yum – это пакетный менеджер Red Hat, который может запрашивать информацию о доступных пакетах, загружать пакеты из репозитория, устанавливать и удалять их, обновлять всю систему до последней доступной версии. Yum выполняет автоматическое разрешение зависимостей, при установке, удалении или обновлении пакетов.

Yum может быть настроен на работу с дополнительными репозиториями или источниками пакетов, а также предоставляет множество плагинов, которые улучшают и расширяют его возможности. Yum во многом повторяет функционал RPM, дополнительно, также многие опции командной строки похожи друг на друга. Yum позволяет легко и быстро управлять пакетами на одной машине или на группе машин.

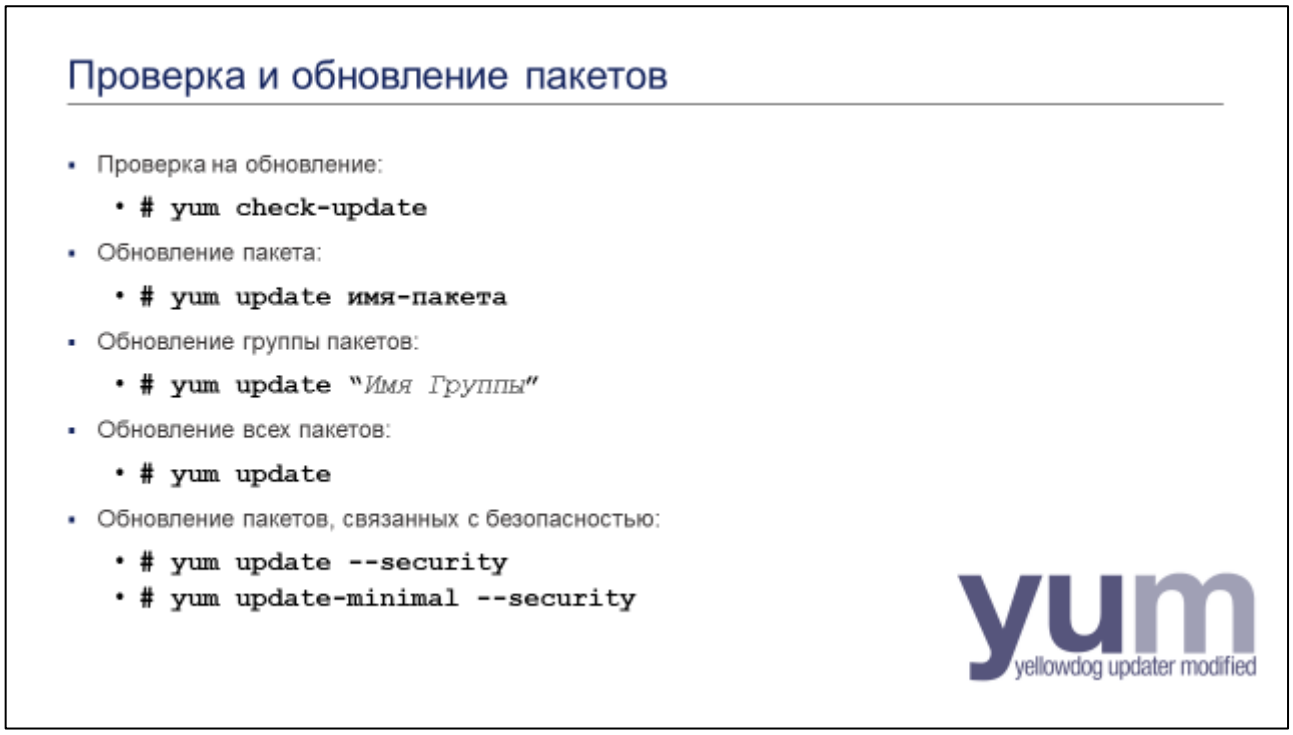
ПРИМЕЧАНИЕ

Yum предоставляет безопасное управление пакетами за счет включения проверки подписей GPG (Gnu Privacy Guard, также известный как GnuPG) на пакетах подписанных при помощи GPG. Проверка подписей GPG может быть включена как на всех репозиториях так и на отдельных из них. Это обозначает, что вы доверяете RPM-пакету, скачанному из репозитория, убедившись в том, что он не был модифицирован.

Yum также позволяет вам настроить ваш собственный репозиторий пакетов для загрузки и установки на прочих машинах. По возможности Yum использует параллельную загрузку нескольких пакетов и метаданных для увеличения скорости загрузки.

ПРИМЕЧАНИЕ

Для установки, удаления или обновления пакетов на системе при помощи yum необходимо иметь привилегии супер-пользователя. Для получения необходимых прав можно воспользоваться **su** или **sudo**.



Yum позволяет проверить наличие доступных обновлений для вашей системы. Вы можете вывести список обновлений пакетов доступных для вашей системы и обновить их все, либо выборочно обновить отдельные пакеты.

Вы можете обновить один пакет, несколько пакетов или все пакеты сразу. Если у зависимых пакетов обновляемого пакета присутствуют обновления, они будут обновлены автоматически.

Обновление одного пакета.

Для обновления оного пакета, выполните следующую команду **yum update** *имя_пакета* от имени пользователя root.

Пример обновления пакета **rpm**:

```
# yum update rpm
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Resolving Dependencies
--> Running transaction check
---> Package rpm.x86_64 0:4.11.1-25.el7 will be updated
--> Processing Dependency: rpm = 4.11.1-25.el7 for package: rpm-python-4.11.1-
25.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-25.el7 for package: rpm-libs-4.11.1-
25.el7.x86_64
---> Package rpm.x86_64 0:4.11.3-17.el7 will be an update
…
---> Package rpm-build-libs.x86_64 0:4.11.3-17.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

```
================================================================================
 Package                Arch           Version                Repository    Size
================================================================================
Updating:
 rpm                x86_64         4.11.3-17.el7         base         1.2 M
Updating for dependencies:
 rpm-build-libs     x86_64         4.11.3-17.el7         base         103 k
 rpm-libs           x86_64         4.11.3-17.el7         base         272 k
 rpm-python         x86_64         4.11.3-17.el7         base          79 k

Transaction Summary
================================================================================
Upgrade  1 Package (+3 Dependent packages)

Total download size: 1.6 M
Is this ok [y/d/N]:
```

Вывод содержит следующие элементы:

6. **Loaded plugins: fastestmirror, langpacks** – yum всегда информирует о том, какие плагины установлены и включены.
7. **Resolving Dependencies** – Yum использует транзакционную проверку для разрешения зависимостей.
8. Yum представляет информацию об обновлении и затем запрашивает подтверждение о начале процесса обновления, по умолчанию yum запускается в интерактивном режиме. Если вы уже знаете о том, что будет делать команда yum, вы можете воспользоваться опциональным ключом **-y**, чтобы автоматически отвечать yes на все запросы, которые выдаст yum (в данном случае yum запускается не интерактивно). Однако, вы должны быть всегда в курсе о том какие изменения yum собирается внести в систему, это позволит проще предотвращать возможные неисправности. Вы также можете выбрать загрузку пакетов без установки, для этого необходимо выбрать опцию **d** в строке запроса загрузки. Это запустит фоновый процесс загрузку выбранного пакета.

Аналогичным образом вы может обновить группу пакетов. Для этого воспользуйтесь командой **yum group update** *Имя Группы*, от имени пользователя root:

```
# yum group update Development Tools
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Warning: group Tools does not exist.
Resolving Dependencies
--> Running transaction check
---> Package autoconf.noarch 0:2.69-11.el7 will be installed
…
```

Yum также поддерживает команду **upgrade**, которая эквивалентна команде update с включенной опцией настройки **obsolete**. По умолчанию опция **obsolete** включена в **/etc/yum.conf**, что делает эти две команды эквивалентными.

ПРИМЕЧАНИЕ

Подробная информация о группах пакетов и опциях настройки представлена дальше в модуле.

Обновление всех пакетов и их зависимостей.

Для обновления всех пакетов и их зависимостей введите команду **yum update** без каких-либо опций.

```
# yum update
```

Обновление пакетов, связанных с безопасностью.

Обнаружение пакетов, имеющих обновления безопасности и последующее обновление этих пакетов максимально быстро крайне важно. Для этих целей yum предоставляет специальный плагин. Плагин **security** расширяет возможности команды **yum** при помощи удобного набора команд, подкоманд и опций, связанных с безопасностью.

ПРИМЕЧАНИЕ

Подробная информация о плагинах представлена дальше в модуле.

Для обновления пакета, до последней версии, если у него было обновление безопасности, используйте следующую команду:

```
# yum update --security
```

Для обновления пакета, до версии с обновлением безопасности, используйте следующую команду:

```
# yum update-minimal --security
```

Рассмотрим пример:

- Пакет **kernel-4.1.6-1** установлен на системе.
- Пакет **kernel-4.1.6-2** содержит обновление безопасности.
- Пакет **kernel-4.1.6-3** содержит исправление ошибки.

Команда yum update-minimal --security, обновит пакет до **kernel-4.1.6-2**, а команда **yum update --security** обновит пакет до **kernel-4.1.6-3**.

Предотвращение изменений конфигурационных файлов.

Вы неизбежно вносите изменения в конфигурационные файлы, установленные вместе с пакетами в системе. RPM, который использует YUM для внесения изменений в систему, предоставляет механизм проверки и сохранения целостности изменений.

Вы можете искать все имена, описания и сводную информацию об RPM-пакетах при помощи следующей команды:

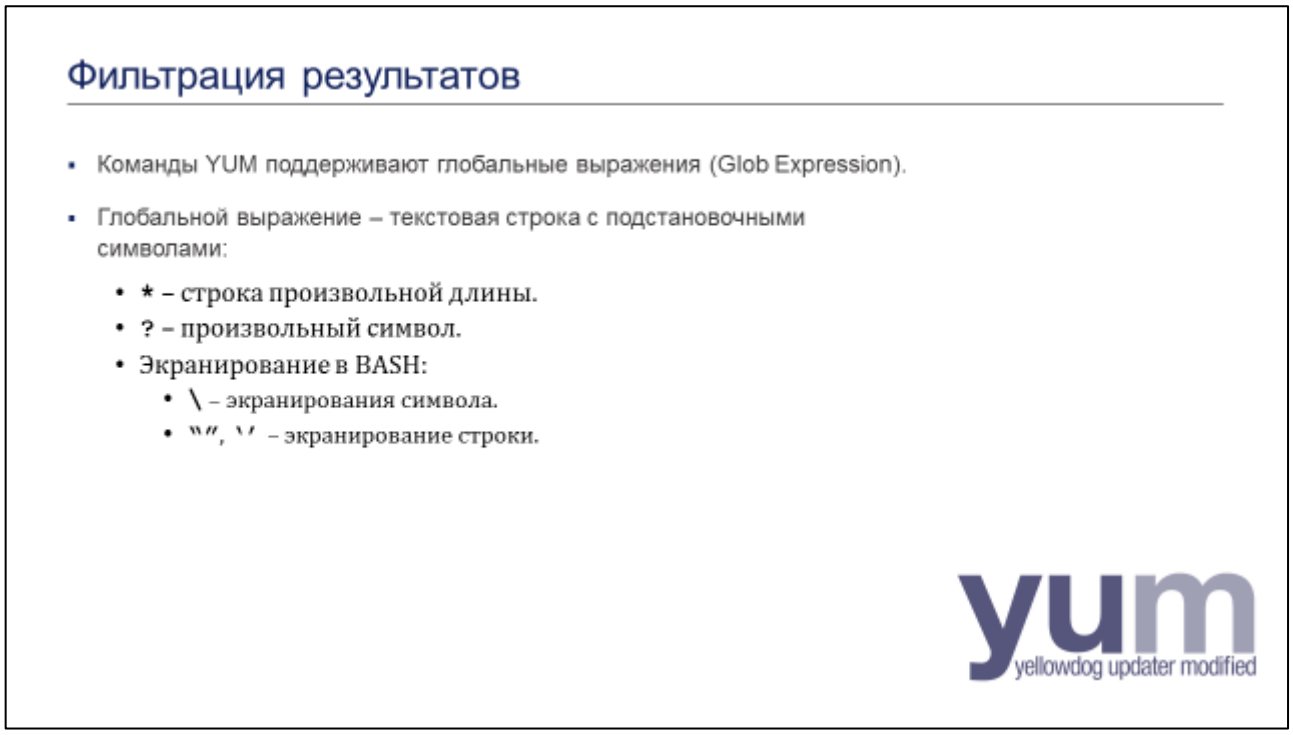
```
yum search ТЕРМИН …
```

Этак команда отображает все совпадения с искомым значением или значениями.

Пример поиска пакетов, которые соответствуют терминам «**rpm**» или «**tool**»:

```
# yum search rpm tool
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
========================= N/S matched: rpm, tool =========================
cabal-rpm.x86_64 : RPM packaging tool for Haskell Cabal-based packages
rpmconf.noarch : Tool to handle rpmnew and rpmsave files
rpmdevtools.noarch : RPM Development Tools
rpmlint.noarch : Tool for checking common errors in RPM packages
rpmreaper.x86_64 : A tool for removing packages from system
rpmrebuild.noarch : A tool to build rpm file from rpm database
tito.noarch : A tool for managing rpm based git projects

  Full name and summary matches only, use "search all" for everything.
```

Команда **yum search** полезна для поиска пакетов, имя которых вам не известно, но вы знаете связанный термин. Учтите, что по умолчанию **yum search** возвращает совпадения в имени пакета или сводке, что делает поиск более быстрым. Используйте команд **yum search all** для исчерпывающего и более длительного поиска.

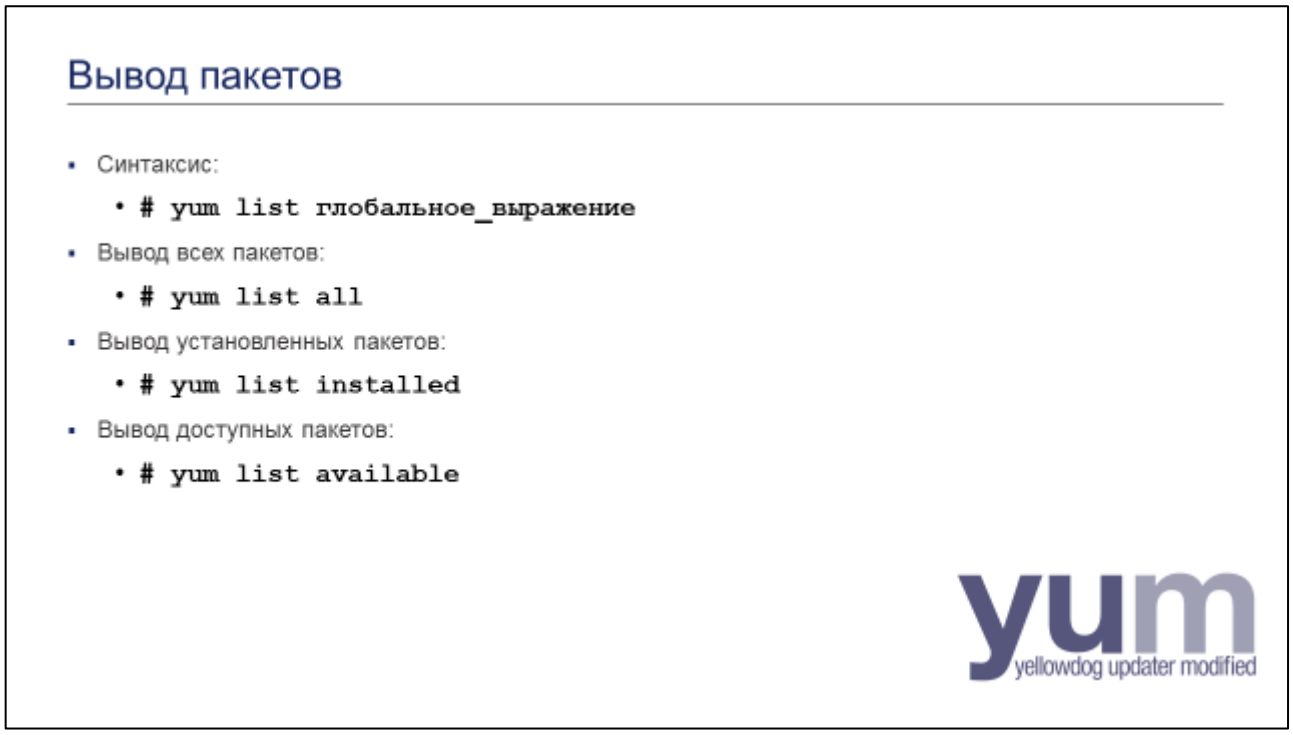


Все команды Yum, возвращающие список значений поддерживают фильтрацию результатов при помощи одного или нескольких глобальных выражений (Glob Expression) в качестве аргументов. Глобальные выражения – это обычные строки символов, которые содержат один или более символов подстановки ***** (обозначает строку произвольной длины) и **?** (обозначает один произвольный символ).

Будьте осторожны при передаче подстановочных символов, так как в первую очередь их обрабатывает Bash. Для передачи глобальных выражений используйте следующие методы:

- Для передачи *****, поставьте перед ним обратный слеш (\).
- Заключите все глобальное выражение в двойные или одинарные кавычки.

Далее в примерах демонстрируется оба этих метода.



Для вывода списка всех установленных и доступных пакетов используйте следующую команду:

```
# yum list all
```

Для вывода установленных и доступных пакетов, которые соответствуют введенному глобальному выражению, используйте следующую команду:

```
yum list глобальное_выражение…
```

Пример использования глобального выражения, указывающего на все пакеты, начинающиеся на **rpm-b**:

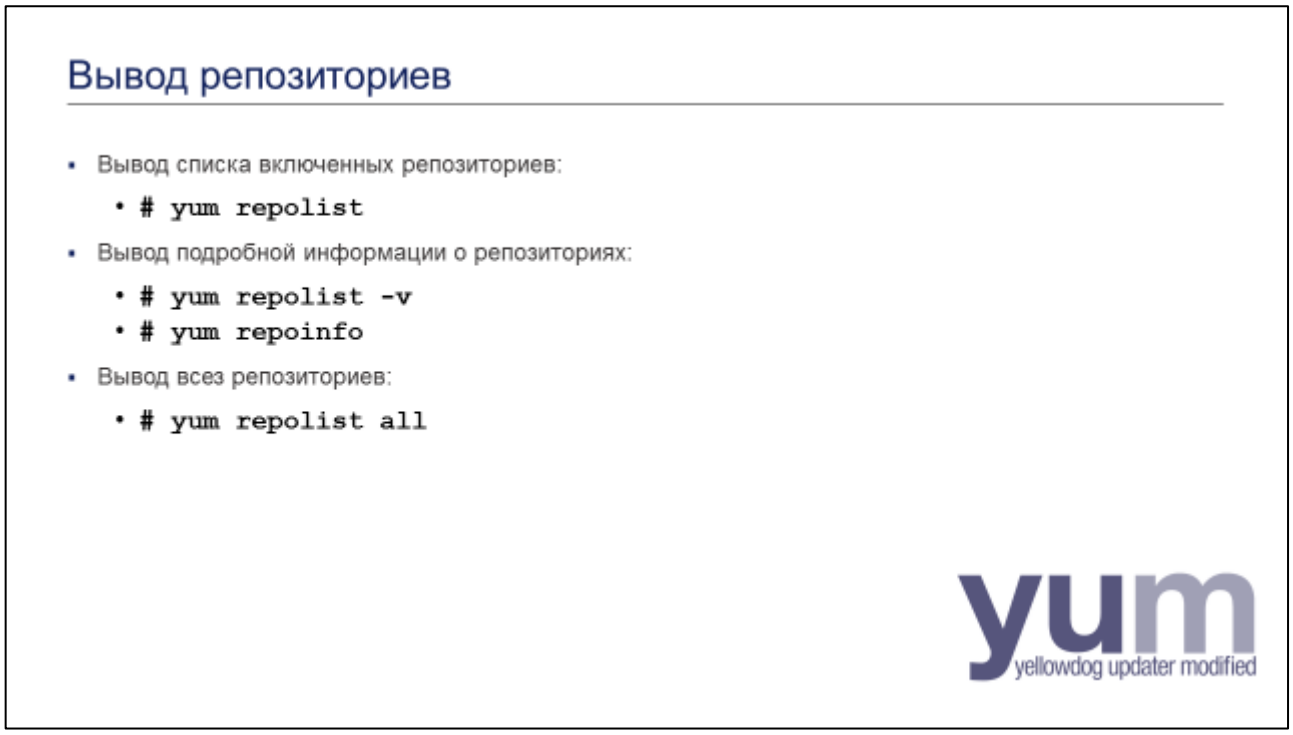
```
# yum list rpm-b\*
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Installed Packages
rpm-build-libs.x86_64                    4.11.1-25.el7                    @base
Available Packages
rpm-build.x86_64                         4.11.3-17.el7                    base
rpm-build-libs.i686                      4.11.3-17.el7                    base
rpm-build-libs.x86_64                    4.11.3-17.el7                    base
```

Для вывода всех установленных на системе пакетов используйте ключевое слово **installed**. Соответствующий список пакетов будет возвращен:

```
# yum list installed rpm-b\*
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Installed Packages
rpm-build-libs.x86_64                    4.11.1-25.el7                    @base
```

Для вывода списка всех пакетов, доступных для установки из репозиториев, используйте ключевое слово **availbale**:

```
# yum list available rpm-b\*
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Available Packages
rpm-build.x86_64                          4.11.3-17.el7                     base
rpm-build-libs.i686                       4.11.3-17.el7                     base
rpm-build-libs.x86_64                     4.11.3-17.el7                     base
```



Для отображения идентификатора, имени и числа пакетов во всех включенных репозиториях используйте следующую команду:

```
# yum repolist
```

Для отображения более подробной информации о репозиториях используйте опцию **–v**. При использовании этой опции, информация будет включать имя файла, общий размер, дату последнего обновления и базовый URL для каждого репозитория в списке. В качестве альтернативы, можно использовать команду repoinfo, которая имеет аналогичный вывод:

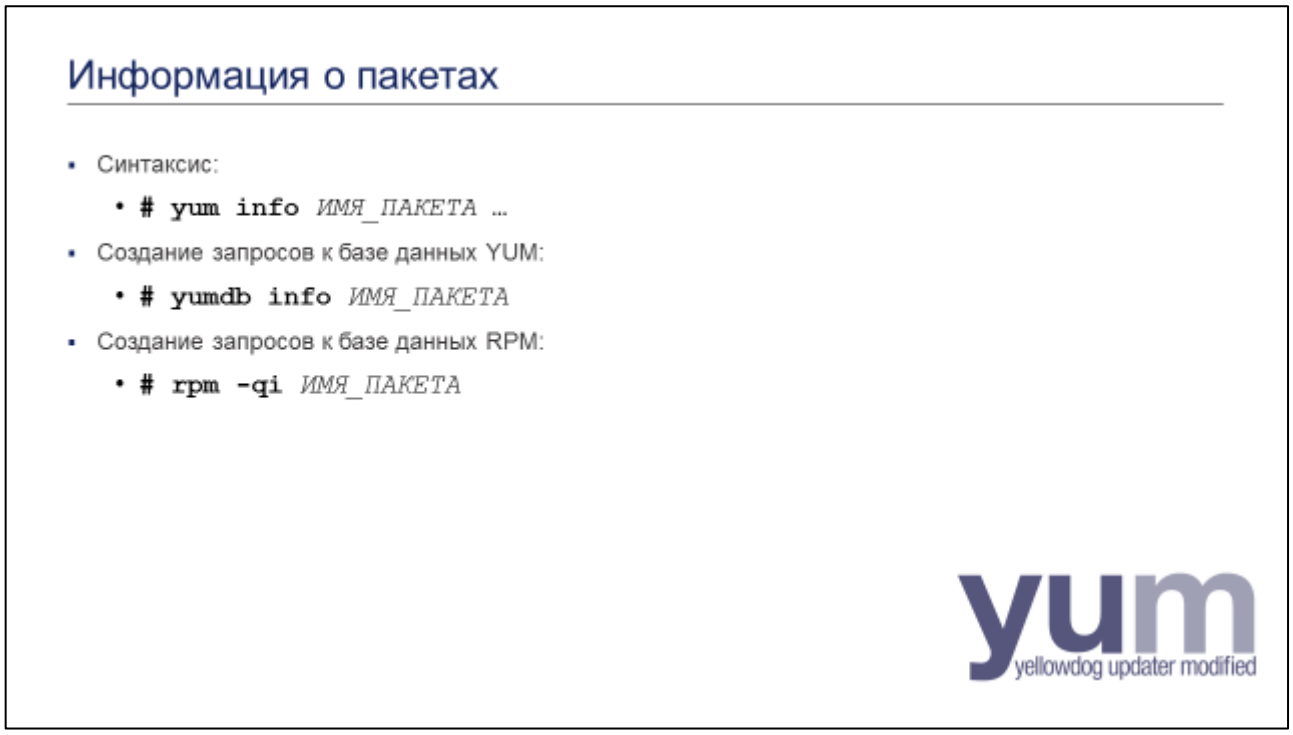
```
# yum repolist –v
# yum repoinfo
```

Для вывода списка всех и включенных, и отключенных репозиториев, используйте следующую команду:

```
# yum repolist all
```

Колонка со статусом (Status) будет добавлена в список вывода, отображая какие репозитории включены.

Передав **disabled**, в качестве первого аргумента, вы сократите вывод до отключенных репозиториев. Можно также передавать идентификатор или имя репозитория или глобальное выражение в качестве аргумента. Обратите внимание, что используется полное соответствие между именем и идентификатором введенного аргумента и репозиторий будет отображен, даже если он не соответствует фильтру **enabled** или **disabled**.



Для отображения информации об одном или нескольких пакетах, используйте следующую команду (глобальные выражения также поддерживаются).

```
yum info ИМЯ_ПАКЕТА…
```

Пример отображения информации о пакете rpm:

```
# yum info rpm
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Installed Packages
Name        : rpm
Arch        : x86_64
Version     : 4.11.1
Release     : 25.el7
Size        : 2.5 M
Repo        : installed
From repo   : base
Summary     : The RPM package management system
URL         : http://www.rpm.org/
License     : GPLv2+
Description : The RPM Package Manager (RPM) is a powerful command line driven
            : package management system capable of installing, uninstalling,
            : verifying, querying, and updating software packages. Each software
            : package consists of an archive of files along with information
            : about the package like its version, a description, etc.

Available Packages
Name        : rpm
Arch        : x86_64
Version     : 4.11.3
Release     : 17.el7
Size        : 1.2 M
Repo        : base/7/x86_64
```

```
Summary     : The RPM package management system
URL         : http://www.rpm.org/
License     : GPLv2+
Description : The RPM Package Manager (RPM) is a powerful command line driven
            : package management system capable of installing, uninstalling,
            : verifying, querying, and updating software packages. Each software
            : package consists of an archive of files along with information
            : about the package like its version, a description, etc.
```

Команда **yum info** *ИМЯ_ПАКЕТА* похожа на команду **rpm -q -info** *ИМЯ_ПАКЕТА*, но предоставляет дополнительную информацию: идентификатор репозитория Yum, в котором был найден RPM-пакет (строка **From repo:** в выводе).

Использование yumdb.

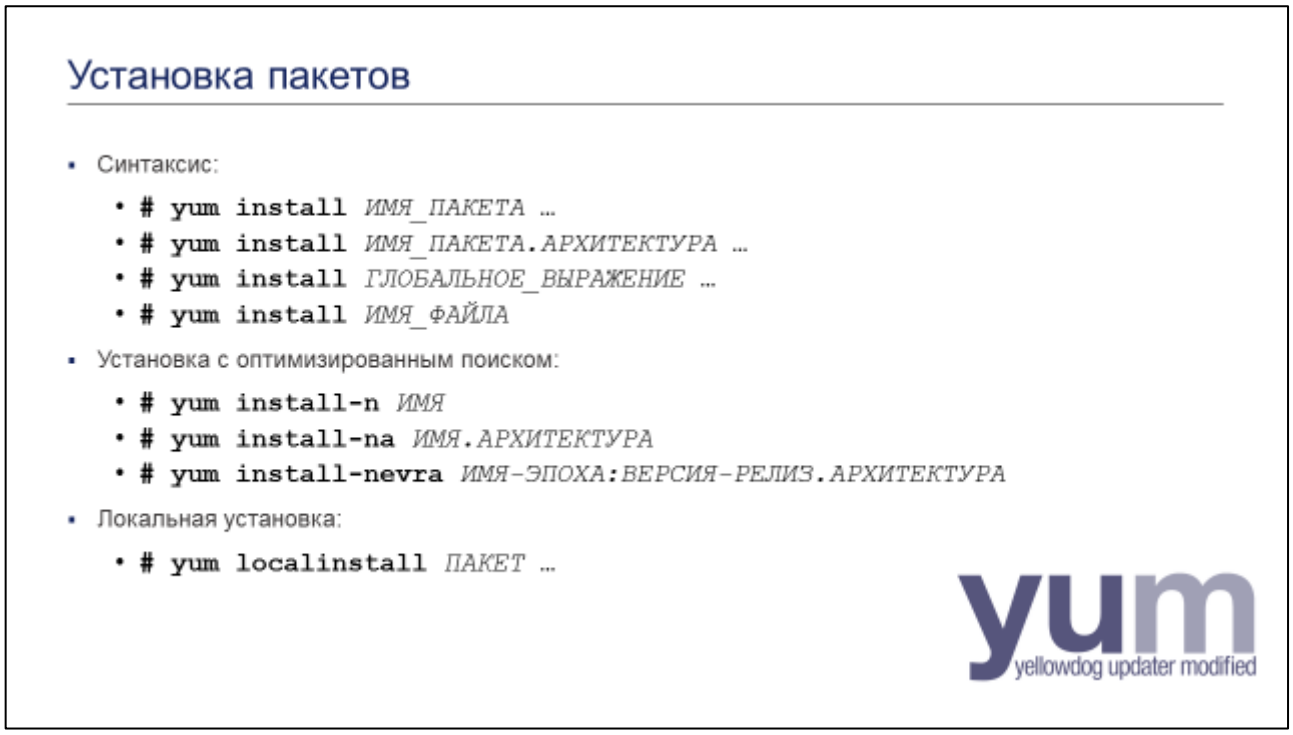
Имеется возможность создавать запросы к базе данных Yum, в качестве альтернативного и полезного источника информации о пакетах при помощи следующей команды:

```
yumdb info имя_пакета
```

Эта команда предоставляет дополнительную информацию о пакете, в том числе контрольную сумму пакета (и алгоритм, использованный для вычисления, такой как SHA-256), команду, которая была использована для установки пакета, цель установки пакета в системе (где user обозначает установку пользователем, а dep обозначает установку в качестве зависимости).

```
# yumdb info rpm
Loaded plugins: fastestmirror, langpacks
rpm-4.11.1-25.el7.x86_64
     changed_by = 0
     checksum_data =
43671fd5e1073e3589058a24efbbab9139384893da8d7f2e7502d48181f53e9d
     checksum_type = sha256
     command_line = update
     from_repo = base
     from_repo_revision = 1427842153
     from_repo_timestamp = 1427842246
     installed_by = 4294967295
     origin_url = http://mirror.yandex.ru/centos/7.1.1503/os/x86_64/Packages/rpm-
4.11.1-25.el7.x86_64.rpm
     reason = user
     releasever = 7
     var_uuid = 777cc08f-59ee-4df5-b1b5-00e24477a0af
```

Для получения дополнительной информации по команде **yumdb**, обратитесь к странице руководства **yumdb(8)**.



Для установки отдельного пакета и всех его неустановленных зависимостей, введите следующую команду от имени пользователя root:

```
yum install имя_пакета
```

Вы можете установить несколько пакетов, добавляя их имена в качестве аргументов:

```
yum install имя_пакета имя_пакета …
```

Если вы устанавливаете пакет на системе с несколькими наборами библиотек (Multilib), таких как машины с архитектурой AMD64 или Intel64 вы можете указать архитектуру пакета (если она доступна в репозитории), добавив `.архитектура` к имени пакета.

```
yum insyall имя_пакета.архитектура
```

Пример установки Wireshark с архитектурой **i686** в RedOS 7 (x64):

```
# yum install wireshark.i686
```

Также можно использовать глобальные выражения (Glob Expressions) при установке нескольких идентичных пакетов

```
yum install глобальное_выражение …
```

Пример установки всех пакетов, имя которых начинается на **openssl**:

```
# yum install openssl\*
```

В дополнение к имени пакета и глобальным выражениям вы можете передавать имя файла команде yum install. Если вы знаете имя бинарного файла, который хотите установить, но не знаете имя пакета, вы можете указать абсолютный путь для команды yum install:

```
# yum install /usr/sbin/named
```

В таком случае yum ищет в своем списке пакет, который содержит файл **/usr/sbin/named**, при нахождении запрашивает у вас необходимость установки.

Как можно видеть из примера выше, команда **yum install** не требует прямого указания аргументов. Он может обрабатывать различные форматы имен пакетов и глобальных выражений, которые упрощают установку для пользователей. С другой стороны, это требует некоторого времени, чтобы yum распознал ввод корректно, особенно если вы указали большое количество пакетов. Для оптимизации поиска пакетов, вы можете использовать следующие команды, для прямого указания, как распознавать аргументы:

```
yum install-n имя
yum install-na имя.архитектура
yum install-nevra имя-эпоха:версия-релиз.архитектура
```

При помощи команды **install-n** указывается на прямое соответствие имени пакета. Команда **install-na** указывает, что последующие аргументы содержат имя и архитектуру разделенные символом точки. Команды **install-nevra**, **yum** указывает, что последующие аргументы имеют следующую форму: *имя-эпоха:версия-релиз.архитектура*. Аналогично вы можете использовать команды yum rmove-n, yum rmove-na и yum rmove-nevra для поиска пакетов на удаление.

ПРИМЕЧАНИЕ

Если вы хотите установить пакет, который содержит бинарный файл named, но не знаете в каком каталоге **bin** или **sbin** он должен находиться, используйте команду **yum provides** с глобальным выражением:

```
# yum provides *bin/named
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
32:bind-9.9.4-29.el7.x86_64 : The Berkeley Internet Name
Domain (BIND) DNS
                            : (Domain Name System) server
Repo        : base
Matched from:
Filename    : /usr/sbin/named
…
```

По аналогии можно использовать команду следующего вида:

```
yum provides "*/имя_файла"
```

Следующий пример предоставляет обзор установки Apache HTTP Server при помощи yum. Для загрузки и установки последней версии **httpd**, используйте следующую команду:

```
# yum install httpd
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-31.el7.centos will be updated
```

```
---> Package httpd.x86_64 0:2.4.6-40.el7.centos.1 will be an update
--> Processing Dependency: httpd-tools = 2.4.6-40.el7.centos.1 for package: httpd-
2.4.6-40.el7.centos.1.x86_64
--> Running transaction check
---> Package httpd-tools.x86_64 0:2.4.6-31.el7.centos will be updated
---> Package httpd-tools.x86_64 0:2.4.6-40.el7.centos.1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

После выполнения выше указанной команды, yum загружает необходимые плагины и выполняет транзакционную проверку. В данном примере, **`httpd`** уже установлен. Так как установленный пакет старше последнего доступного пакета, он будет обновлен. Тоже самое касается и пакета **`httpd-tools`**. Затем отображается результат транзакции:

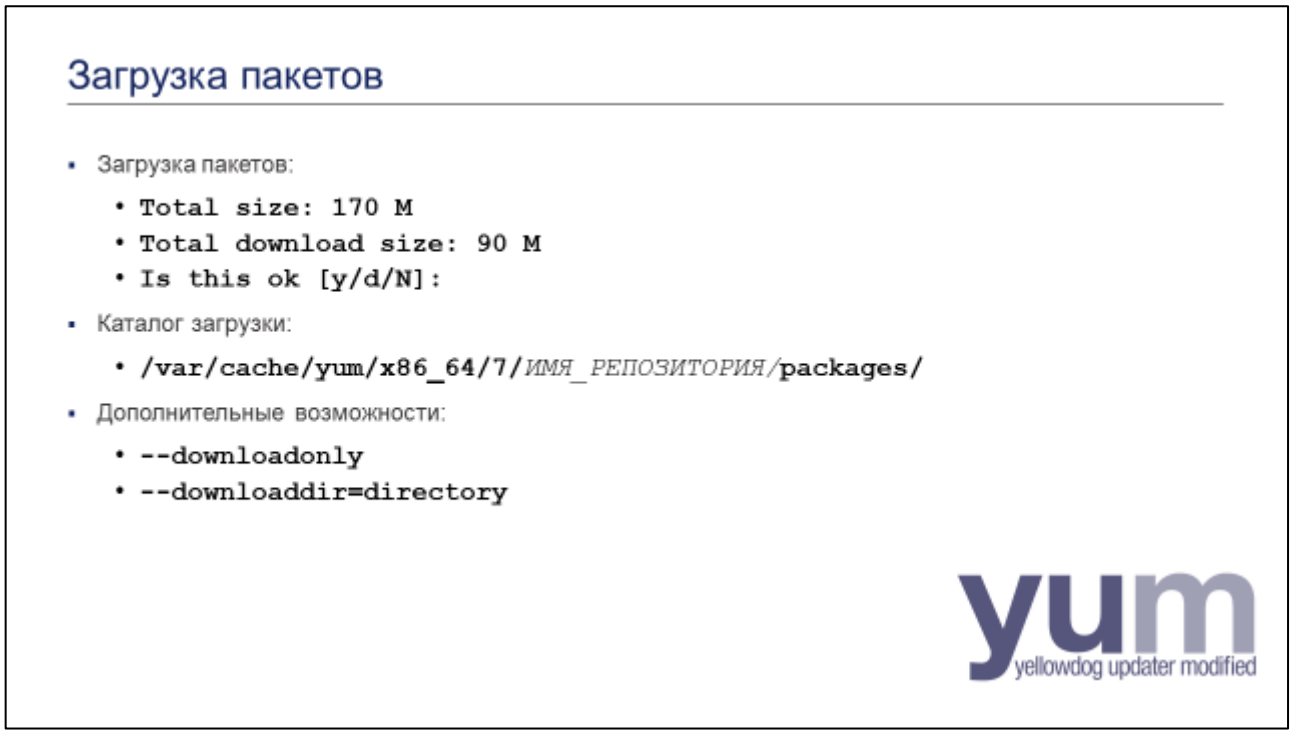
```
================================================================================
 Package            Arch         Version                      Repository     Size
================================================================================
Updating:
 httpd              x86_64       2.4.6-40.el7.centos.1        updates       2.7 M
Updating for dependencies:
 httpd-tools        x86_64       2.4.6-40.el7.centos.1        updates        82 k

Transaction Summary
================================================================================
Upgrade  1 Package (+1 Dependent package)

Total download size: 2.8 M
Is this ok [y/d/N]:
```

На этом шаге yum запрашивает у вас подтверждение установки. Вне зависимости от опций **`y`** (да) и **`N`** (нет), вы можете выбрать **`d`** (только загрузка) для загрузки, но не установки пакетов. Если вы выберите y, процесс установки продолжится и выдаст следующие сообщения:

```
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
(1/2): httpd-tools-2.4.6-40.el7.centos.1.x86_64.rpm        |  82 kB   00:00
(2/2): httpd-2.4.6-40.el7.centos.1.x86_64.rpm              | 2.7 MB   00:02
--------------------------------------------------------------------------------
Total                                           1.1 MB/s | 2.8 MB  00:02
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : httpd-tools-2.4.6-40.el7.centos.1.x86_64                     1/4
  Updating   : httpd-2.4.6-40.el7.centos.1.x86_64                           2/4
  Cleanup    : httpd-2.4.6-31.el7.centos.x86_64                             3/4
  Cleanup    : httpd-tools-2.4.6-31.el7.centos.x86_64                       4/4
  Verifying  : httpd-2.4.6-40.el7.centos.1.x86_64                           1/4
  Verifying  : httpd-tools-2.4.6-40.el7.centos.1.x86_64                     2/4
  Verifying  : httpd-tools-2.4.6-31.el7.centos.x86_64                       3/4
  Verifying  : httpd-2.4.6-31.el7.centos.x86_64                             4/4

Updated:
  httpd.x86_64 0:2.4.6-40.el7.centos.1

Dependency Updated:
  httpd-tools.x86_64 0:2.4.6-40.el7.centos.1

Complete!
```

Для установки предварительно загруженных пакетов из локального каталога на вашей системе, используйте следующую команду:

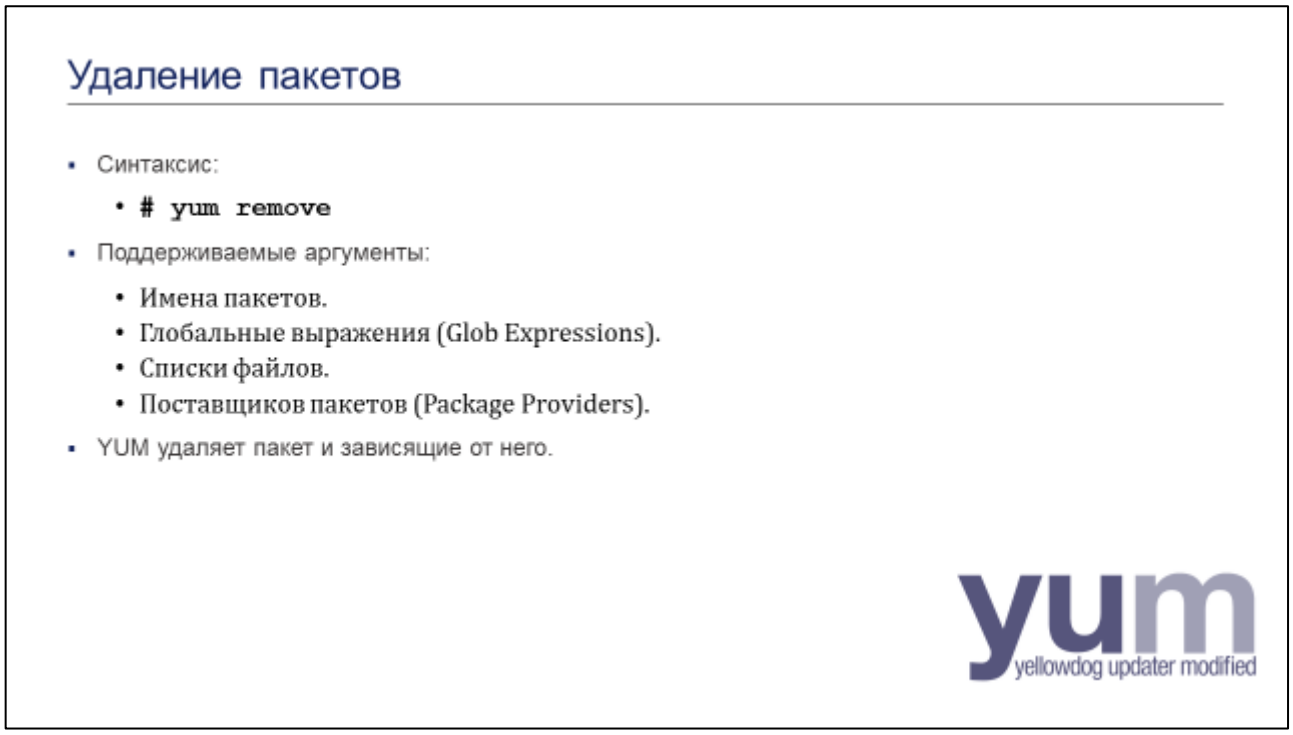
```
yum localinstall путь
```



Как было показано в предыдущем примере, в процессе установки пакетов у вас запрашивается подтверждение установки:

```
Total download size: 2.8 M
Is this ok [y/d/N]:
```

При помощи опции **d**, **yum** скачивает пакеты без установки. Вы можете позже установить эти пакеты в режиме офлайн, при помощи команды **yum localinstall** или вы можете распространить их между другими устройствами. Загруженные пакеты сохраняются в одном из подкаталогов каталога кэширования, по умолчанию: **/var/cache/yum/***архитектура***/***релиз***/packages**. Загрузка может выполняться в фоновом режиме, таким образом, оставляя возможность параллельно использовать **yum** для других операций.



Подобно установке пакетов, yum позволяет их удалять. Для удаления пакета, также, как и зависящих от него, выполните следующую команду от имени пользователя root:

```
yum remove имя_пакета…
```

Пример удаления пакета **proftpd**:

```
# yum remove proftpd
Loaded plugins: fastestmirror, langpacks
Resolving Dependencies
--> Running transaction check
---> Package proftpd.x86_64 0:1.3.5b-1.el7 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch            Version                   Repository      Size
================================================================================
Removing:
 proftpd          x86_64          1.3.5b-1.el7              @epel          9.7 M

Transaction Summary
================================================================================
Remove  1 Package

Installed size: 9.7 M
Is this ok [y/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Erasing    : proftpd-1.3.5b-1.el7.x86_64                                  1/1
warning: /etc/ftpusers saved as /etc/ftpusers.rpmsave
  Verifying  : proftpd-1.3.5b-1.el7.x86_64                                  1/1

Removed:
```

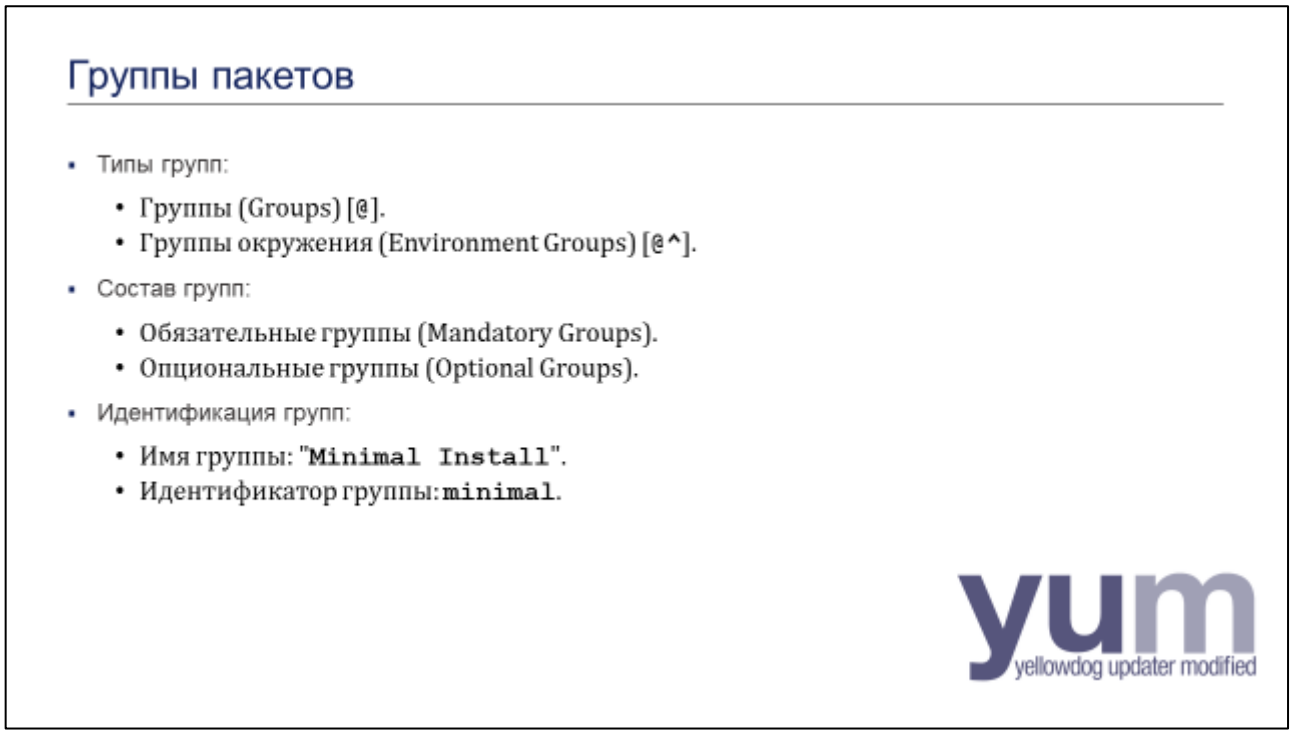
```
  proftpd.x86_64 0:1.3.5b-1.el7

Complete!
```

Подобно тому, как это работает с командой yum install, yum remove может принимать следующие аргументы:

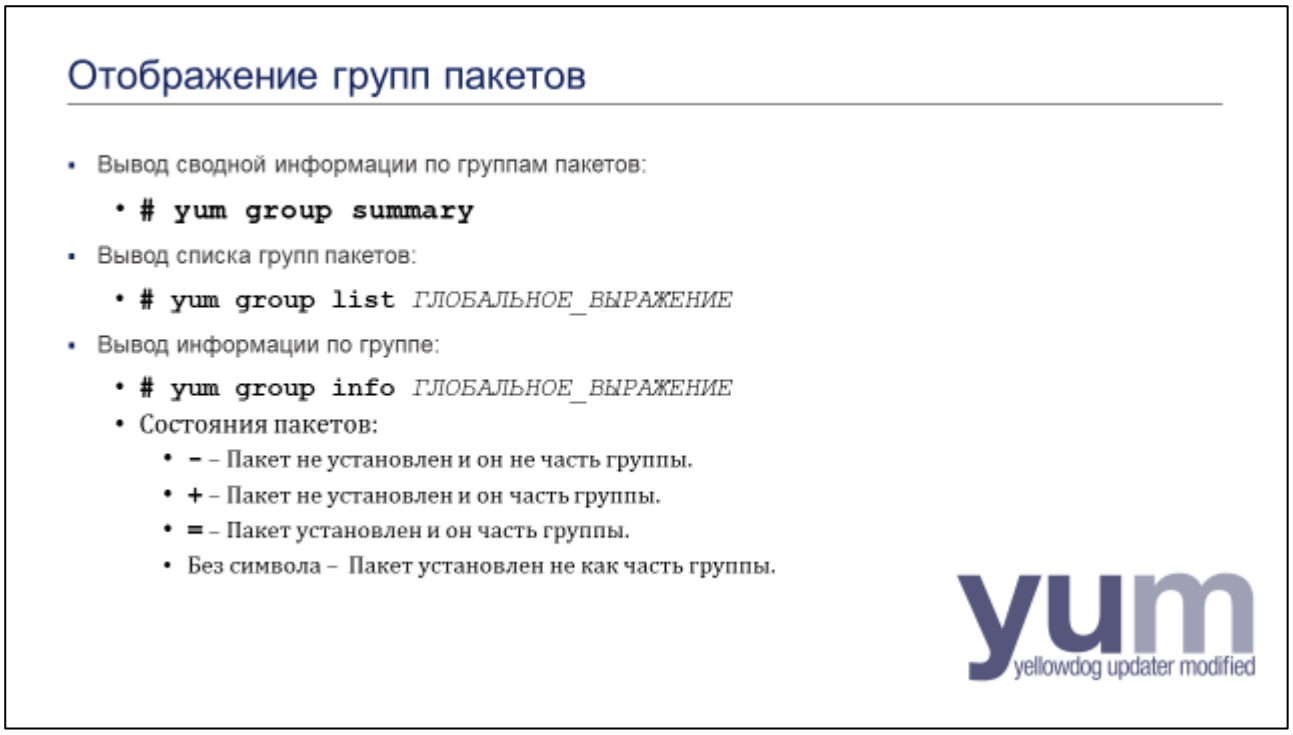
- Имена пакетов.
- Глобальные выражения (Glob Expressions).
- Списки файлов.
- Поставщиков пакетов (Package Providers).

ПРИМЕЧАНИЕ

Yum не может удалить пакет, без удаления, зависящего от него пакета. Это тот тип операций, который может быть выполнен только при помощи RPM, такая возможность не рекомендуется к использованию и потенциально может повлечь за собой некорректную работу вашей системы или аварию.



Группа пакетов (Package Group) – это коллекция пакетов, которые имеют одинаковое назначение, например, **`System Tools`** или **`Sound and Video`**. Установка группы пакетов, выстраивает в очередь набор зависимых пакетов и значительно экономит время. Команда **`yum groups`** – это команда верхнего уровня, позволяющая выполнять все возможные операции над группами пакетов в **`yum`**.



Опция **summary** используется для просмотра количества установленных групп, доступных групп, доступных групп окружения, а также установленных и доступных языковых групп:

```
# yum group summary
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Available environment groups: 11
Available Groups: 38
Done
```

Для отображения всех групп пакетов из репозиториев yum используйте опцию **list**. Также можно фильтровать вывод команды при помощи имен групп.

```
yum group list глобальное_выражение
```

Несколько опциональных аргументов могут быть переданы этой команде, в том числе **hidden** для отображения групп, не помеченных как видимых для пользователя (User Visible) и **ids** для вывода идентификаторов (ID) групп. Также можно добавить опции **language**, **environment**, **installed** и **available** для сокращения вывода команды до указанного типа групп.

Для вывода более подробной информации о группе пакетов, в том числе списков обязательных и опциональных пакетов, если это поддерживается, используйте следующую команду:

```
yum group info глобальное_выражение
```

Пример вывода информации о группе пакетов **Development Tools**:

```
# yum group info 'Development Tools'
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
```

```
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com

Group: Development Tools
 Group-Id: development
 Description: A basic development environment.
 Mandatory Packages:
   +autoconf
   +automake
    binutils
…
   +rpm-sign
 Default Packages:
   +byacc
   +cscope
   +ctags
…
   +systemtap
 Optional Packages:
   ElectricFence
…
   translate-toolkit
```

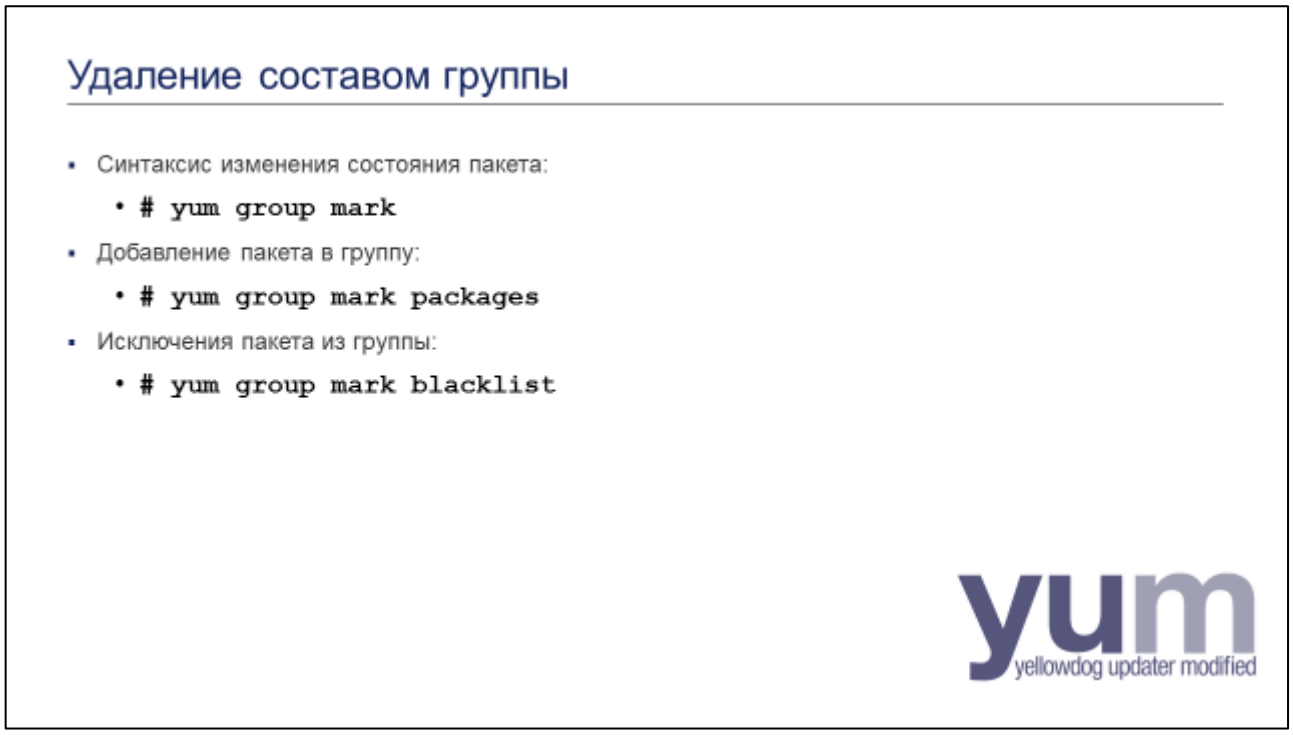
Как можно видеть в примере, у пакетов группы возможны различные состояния, которые маркируются следующими символами:

- **-** – Пакет не установлен, и он не будет установлен, как часть группы пакетов.
- **+** – Пакет не установлен, но он будет установлен, при следующем выполнении yum upgrade или yum group upgrade.
- **=** – Пакет установлен и был установлен как часть группы пакетов.
- Без символа – Пакет установлен на системе, но не в составе группы. Это значит, что он не будет удален командой yum group remove.

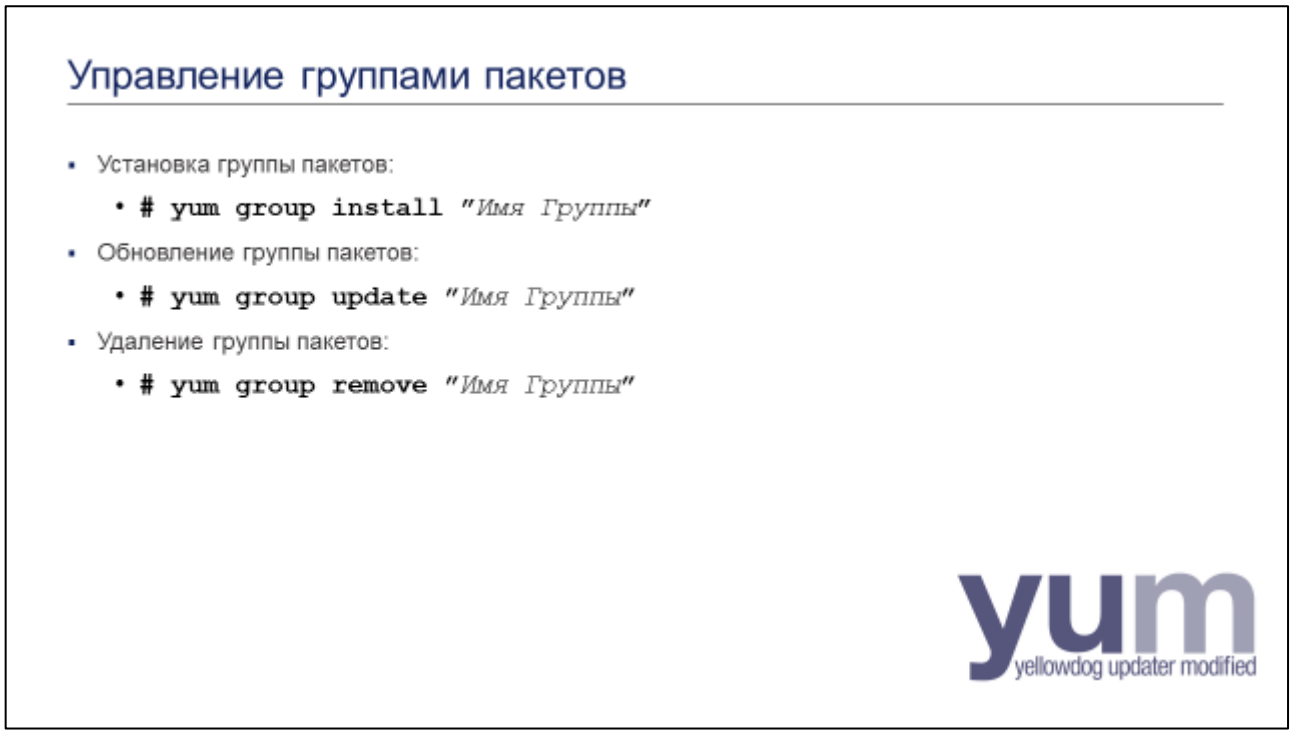
Эти различия имеют значения, только когда параметр настройки group_command выставлен в значение **objects** (параметр по умолчанию для RedOS). Изменение значения этого параметра приведет к отключению отслеживания вхождения пактов в группы.

ПРИМЕЧАНИЕ

Вы можете определить группу окружения при помощи префикса **@^** и группу пакетов при помощи **@**. Когда используется **yum group list**, **info**, **install** или **remove** передайте **@***имя_группы* для указания группы пакетов, **@^***имя_группы* для указания группы окружения или *имя_группы* для указания обоих типов групп.



Вы можете изменить состояния пакета при помощи команды **yum group mark**. Например, **yum group mark packages** сделает любые указанные пакеты членами указанной группы. Для того чтобы избежать установки новых пакетов при обновлении группы, используйте команду **yum group mark blacklist**. Чтобы больше узнать о возможностях **yum group mark**, обратитесь к страницам руководства **yum(8)**.



Установка групп пакетов.

Каждая группа пакетов имя и идентификатор группы (*groupid*). Для вывода имен всех групп пакетов и идентификаторов групп, воспользуйтесь следующей командой:

```
yum group list ids
```

Для поиска имени и идентификатора группы пакетов, связанных с окружением рабочего стола KDE, воспользуйтесь следующей командой:

```
# yum group list ids kde\*
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Available environment groups:
   KDE Plasma Workspaces (kde-desktop-environment)
Done
```

Некоторые группы скрыты параметрами в настроенных репозиториях. Для их отображения можно использовать опцию **hidden**.

```
# yum group list hidden ids kde\*
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: mirror.digitalhusky.com
 * epel: mirror.yandex.ru
 * extras: mirror.digitalhusky.com
 * updates: mirror.digitalhusky.com
Available environment groups:
   KDE Plasma Workspaces (kde-desktop-environment)
Available Groups:
```

```
   KDE (kde-desktop)
   KDE Applications (kde-apps)
   KDE Multimedia Support (kde-media)
Done
```

Для установки группы пакетов, можно использовать ее имя:

```
yum group install "Имя группы"
```

Также для установки можно использовать идентификатор группы (*groupid*):

```
yum group install groupid
```

Если вы используете символ **@** для указания группы пакетов или **@^** для указания группы окружения, можно опустить слово group при установке:

```
yum install @группа
yum install @^группа
```

Пример четырех способов установки группы пакетов:

```
# yum group install "KDE Plasma Workspaces"
# yum group install kde-desktop-environment
# yum install @^"KDE Plasma Workspaces"
# yum install @^kde-desktop-environment
```

Удаление групп пакетов.

Для удаления группы пакетов используется синтаксис аналогичный **install** с указанием имени группы или идентификатора:

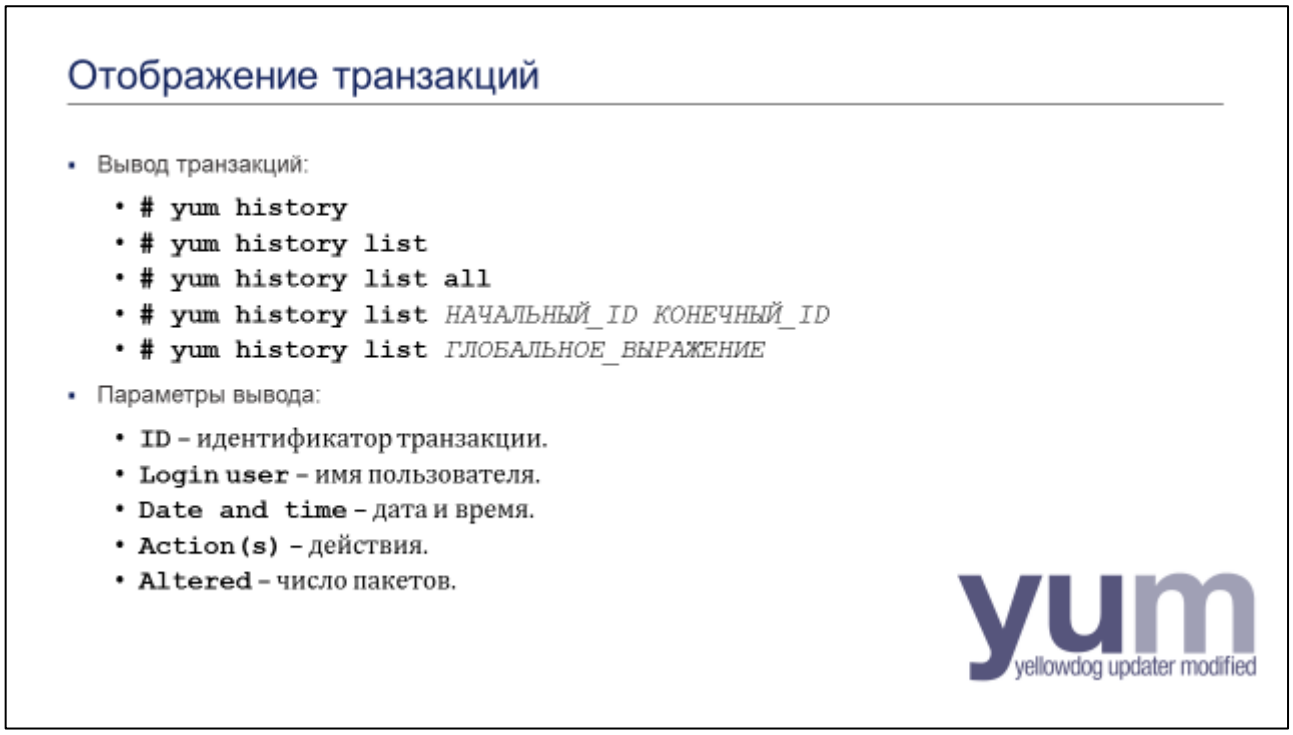
```
yum group remove "Имя группы"
yum group remove groupid
```

Также можно использовать имя группы в кавычках или идентификатор группы с символом **@** (для указания группы пакетов) или **@^** (для указания группы окружения), без использования команды **group**:

```
yum remove @группа
yum remove @^группа
```

Пример четырех способов удаления группы пакетов:

```
# yum group remove "KDE Plasma Workspaces"
# yum group remove kde-desktop-environment
# yum remove @^"KDE Plasma Workspaces"
# yum remove @^kde-desktop-environment
```



Команда **yum history** позволяет пользователям просмотреть информацию о временных шкалах транзакций yum, датах и времени их выполнения, числу задействованных пакетов, когда транзакция успешно выполнилась или была прервана и изменениях в базе данных RPM между транзакциями. Дополнительно, эта команда может быть использована для отмены или повторения указанной транзакции. Все данные истории сохраняются в базе данных истории (History DB) в каталоге **/var/lib/yum/history/**.

Отображение транзакций.

Для отображения списка из двадцати последних транзакций, от имени пользователя root, выполните одну из следующих команд:

```
# yum history
# yum history list
```

Для отображения всех транзакций, добавьте ключевое слово all:

```
# yum history list all
```

Для отображения транзакций из указанного диапазона, используйте следующую форму:

```
# yum history list начальный_ID конечный_ID
```

Также можно отобразить транзакции соответствующую указанному пакету или пакетам. Для этого используйте команд **yum history list** с именем пакета или глобальным выражением (Global Expression):

```
# yum history list глобальное_выражение
```

Пример вывода 5 самых старых транзакций.

В выводе команды yum history list самые последние транзакции отображаются в начале списка. Для отображения информации о пяти самых старых транзакций, хранящихся в базе данных истории, используйте следующую команду:

```
# yum history list 1..5
Loaded plugins: langpacks, product-id, subscription-manager
ID     | Login user             | Date and time    | Action(s)         | Altered
-------------------------------------------------------------------------------
5      | User <user>            | 2013-07-29 15:33 | Install           | 1
4      | User <user>            | 2013-07-21 15:10 | Install           | 1
3      | User <user>            | 2013-07-16 15:27 | I, U              | 73
2      | System <unset>         | 2013-07-16 15:19 | Update            | 1
1      | System <unset>         | 2013-07-16 14:38 | Install           | 1106
history list
```

Все формы команды **yum history list** создают табличный вывод, в котором каждая строка состоит из следующих колонок:

- **ID** – Целочисленное значение, определяющее транзакцию.
- **Login user** –Имя пользователя, чья сессия была использована для начала транзакции. Эта информация обычно предоставляется в форме: **Full Name <***username***>**. Для транзакций, которые были вызваны не пользователем (такие как автоматическое обновление системы) используется **System <unset>**.
- **Date and time** – Дата и время вызова транзакции.
- **Action(s)** – список действий, которые были выполнены в рамках транзакции.
- **Altered** –Число пакетов, которые были задействованы в транзакции, возможно будет сопровождаться дополнительной информацией.

Значения полей

- Возможные значения поля Actions(s):
  - **Downgrade (D)** - пакет был понижен до предыдущей версии.
  - **Erase (E)** - пакет был удален.
  - **Install (I)** - пакет был установлен.
  - **Obsoleting (O)** - пакет был отмечен как устаревший.
  - **Reinstall (R)** - пакет был переустановлен.
  - **Update (U)** - пакет был обновлен.
- Возможные значения поля **Altered**:
  - **<** - база данных **rpmdb** была изменена (не **yum**).
  - **>** - база данных **rpmdb** была изменена (не **yum**).
  - ***** - транзакция не смогла завершиться.
  - **#** - транзакция завершилась (не нулевой выходной код).
  - **E** - транзакция завершилась (ошибка или предупреждение).
  - **P** - транзакция завершилась (проблемы существуют в базе данных **rpmdb**).
  - **S** - транзакция завершилась (использована опция **--skip**)

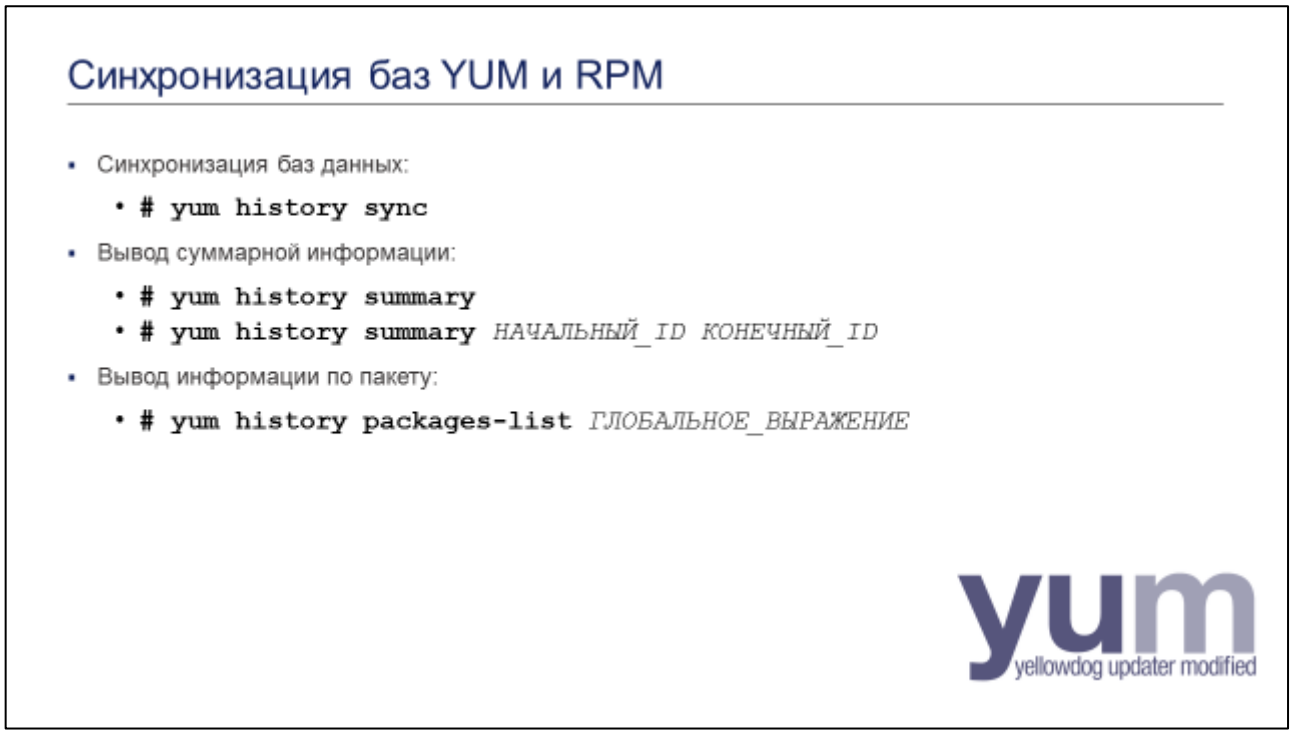
yum
yellowdog updater modified

Возможные значения поля **Action(s)**:

Действие	Аббревиатура	Описание
Downgrade	**D**	Как минимум один пакет был понижен до предыдущей версии.
Erase	**E**	Как минимум один пакет был удален.
Install	**I**	Как минимум один пакет был установлен.
Obsoleting	**O**	Как минимум один пакет был отмечен как устаревший.
Reinstall	**R**	Как минимум один пакет был переустановлен.
Update	**U**	Как минимум один пакет был обновлен.

Возможные значения поля **Altered**:

Символ	Описание
<	Прежде чем транзакция завершилась база данных **rpmdb** была изменена (не **yum**).
>	После завершения транзакции база данных **rpmdb** была изменена (не **yum**).
*****	Транзакция не смогла завершиться.
#	Транзакция успешно завершилась, но **yum** вернул не нулевой выходной код.
E	Транзакция успешно завершилась, но ошибка или предупреждение была отображена.
P	Транзакция успешно завершилась, но проблемы уже существуют в базе данных **rpmdb**.
S	Транзакция успешно завершилась, но была использована опция **--skip**.



Для синхронизации контента базы данных **rpmdb** или **yumdb** для любых установленных пакетов с используемой сейчас **rpmdb** или **yumdb**, введите следующую команду:

```
# yum history sync
```

Для отображения общей статистики из используемой на данный момент базы данных введите следующую команду:

```
# yum history stats
Loaded plugins: langpacks, product-id, subscription-manager
File  : //var/lib/yum/history/history-2012-08-15.sqlite
Size  : 2,766,848
Transactions      : 41
Begin time  : Wed Aug 15 16:18:25 2012
End time    : Wed Feb 27 14:52:30 2013
Counts      :
      NEVRAC      : 2,204
      NEVRA       : 2,204
      NA    : 1,759
      NEVR  : 2,204
      rpm DB      : 2,204
      yum DB      : 2,204
history stats
```

Yum также позволяет отобразить сумму всех последних транзакций. Для этого воспользуйтесь следующую командой:

```
# yum history summary
```

Для отображения транзакций только в указанном диапазоне. Введите следующую команду:

```
# yum history summary начальный_ID..конечный_ID
```

По аналогии с командой **yum history list**, можно отобразить суммарную информацию по транзакциям в которых использовался указанный пакет или пакеты, за счет передачи имени пакета или глобального выражения (Glob Expression):

```
# yum history list глобальное_выражение
```

Пример получения суммарной информации по пяти самым старым транзакциям:

```
# yum history summary 1..5
Loaded plugins: langpacks, product-id, subscription-manager
Login user | Time | Action(s) |
Altered
------------------------------------------------------------------------------
Jaromir ... <jhradilek> | Last day | Install | 1
Jaromir ... <jhradilek> | Last week | Install | 1
Jaromir ... <jhradilek> | Last 2 weeks | I, U | 73
System <unset> | Last 2 weeks | I, U | 1107
history summary
```

Все формы команды **yum history summary** выдают упрощенный табличный вывод, похожий на вывод команды **yum history list**.

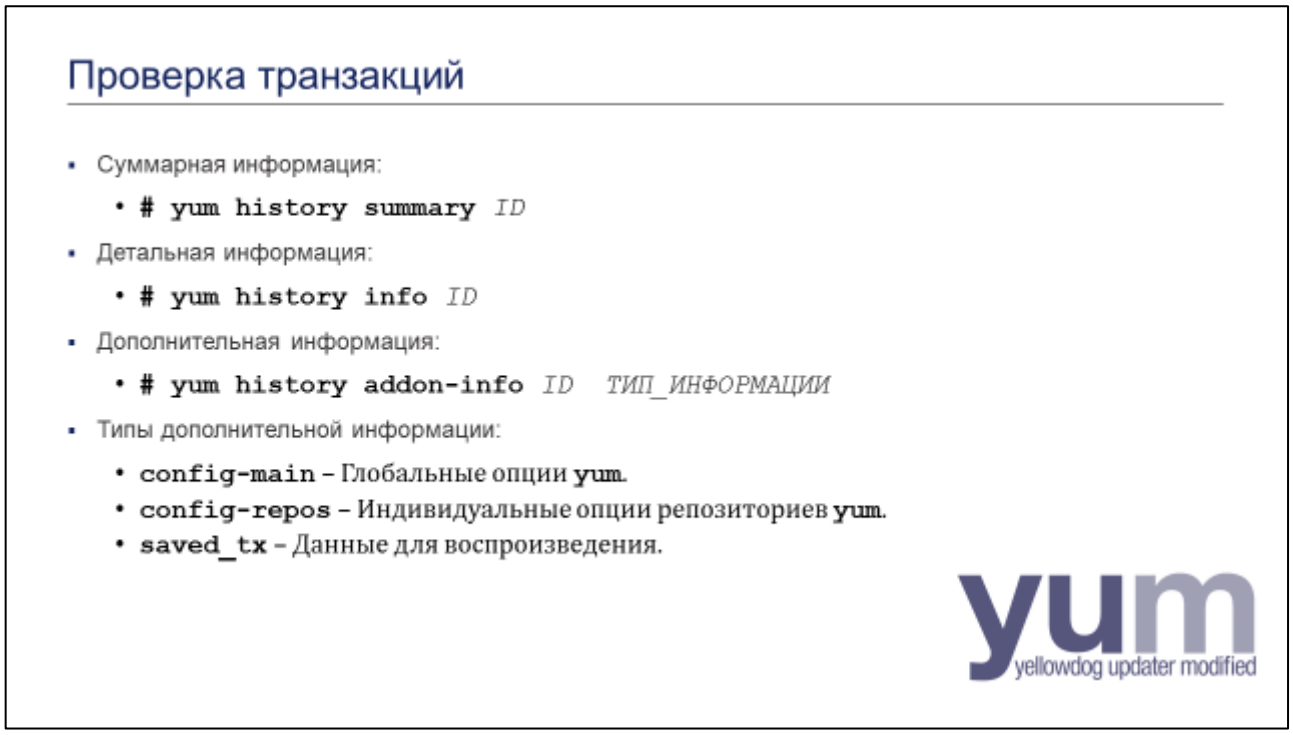
Как показано выше, команды **yum history list** и **yum history summary** ориентированы на транзакции и, несмотря на возможность отобразить только транзакции, связанные с указанным пакетом или пакетами, они не отображают важных подробностей, таких как версии пакетов. Для вывода списка транзакций с точки зрения пакета, выполните следующую команду от имени пользователя root:

```
# yum history package-list глобальное_выражение …
```

Например, для отслеживания пакета **rpm** и связанных пакетов в истории, введите следующую команду:

```
# yum history packages-list rpm\*
Loaded plugins: fastestmirror, langpacks
Repodata is over 2 weeks old. Install yum-cron? Or run: yum makecache fast
ID     | Action(s)      | Package
------------------------------------------------------------------------------
     2 | Updated        | rpm-4.11.1-16.el7.x86_64                           EE
     2 | Update         |     4.11.1-25.el7.x86_64                           EE
     2 | Updated        | rpm-build-libs-4.11.1-16.el7.x86_64                EE
     2 | Update         |                4.11.1-25.el7.x86_64                EE
     2 | Updated        | rpm-libs-4.11.1-16.el7.x86_64                      EE
     2 | Update         |          4.11.1-25.el7.x86_64                      EE
     2 | Updated        | rpm-python-4.11.1-16.el7.x86_64                    EE
     2 | Update         |            4.11.1-25.el7.x86_64                    EE
     1 | Install        | rpm-4.11.1-16.el7.x86_64
     1 | Dep-Install    | rpm-build-libs-4.11.1-16.el7.x86_64
     1 | Dep-Install    | rpm-libs-4.11.1-16.el7.x86_64
     1 | Dep-Install    | rpm-python-4.11.1-16.el7.x86_64
history packages-list
```

В данном примере 4 пакета установлено в процессе начальной установки системы: r**pm-4.11.1-16.el7.x86_64**, **rpm-build-libs-4.11.1-16.el7.x86_64**, **rpm-libs-4.11.1-16.el7.x86_64**, **rpm-python-4.11.1-16.el7.x86_64**. Во второй транзакции все эти пакеты были обновлены.



Для отображения суммарной информации об отдельной транзакции используйте команду **yum history summary** в следующей форме:

```
# yum history summary id
```

Где *id* – это идентификатор транзакции.

Для более детальной проверки транзакции или транзакций, используйте следующую команду от имени пользователя root:

```
# yum history info id …
```

Аргумент *id* опциональный, поэтому, если вы его не указываете, будет выведена информация о последней транзакции. Также для получения информации по нескольким транзакциям вы можете использовать диапазон:

```
# yum history info начальный_id..конечный_id
```

Следующий пример демонстрант вывод двух транзакций:

```
# yum history info 4..5
Loaded plugins: fastestmirror, langpacks
Transaction ID : 4..5
Begin time     : Mon Apr  6 11:11:12 2015
Begin rpmdb    : 484:6156b656808c1ad77d27d0bd16057e74a2b082c4
End time       :         20 08:30:43 2015 (13 days)
End rpmdb      : 490:5d68ce8d29f71639555cd26f668212164c99ce57
User           : root <root>
Return-Code    : Success
Command Line   : install bind-chroot
Command Line   : install httpd
Transaction performed with:
    Installed     rpm-4.11.1-25.el7.x86_64                     @base
    Installed     yum-3.4.3-125.el7.centos.noarch              @base
    Installed     yum-plugin-fastestmirror-1.1.31-29.el7.noarch @base
Packages Altered:
    Dep-Install apr-1.4.8-3.el7.x86_64                 @base
    Dep-Install apr-util-1.5.2-6.el7.x86_64            @base
```

```
    Install     bind-chroot-32:9.9.4-18.el7_1.1.x86_64 @updates
    Install     httpd-2.4.6-31.el7.centos.x86_64       @base
    Dep-Install httpd-tools-2.4.6-31.el7.centos.x86_64 @base
    Dep-Install mailcap-2.1.41-2.el7.noarch            @base
history info
```

Также можно просматривать дополнительную информацию, такую как использованные опции настройки во время транзакции или репозиторий, из которого были установлены пакеты. Для определения доступной информации для определенной транзакции, выполните следующую команду от имени пользователя root:

```
# yum history addon-info id
```

Подобно команде yum history info, когда не указан *id*, yum автоматически использует транзакцию. Другой путь получить последнюю транзакцию – это использовать ключевое слово **last**:

```
# yum history addon-info last
```

Для четвертой транзакции в истории команда yum history addon-info предоставляет следующий вывод:

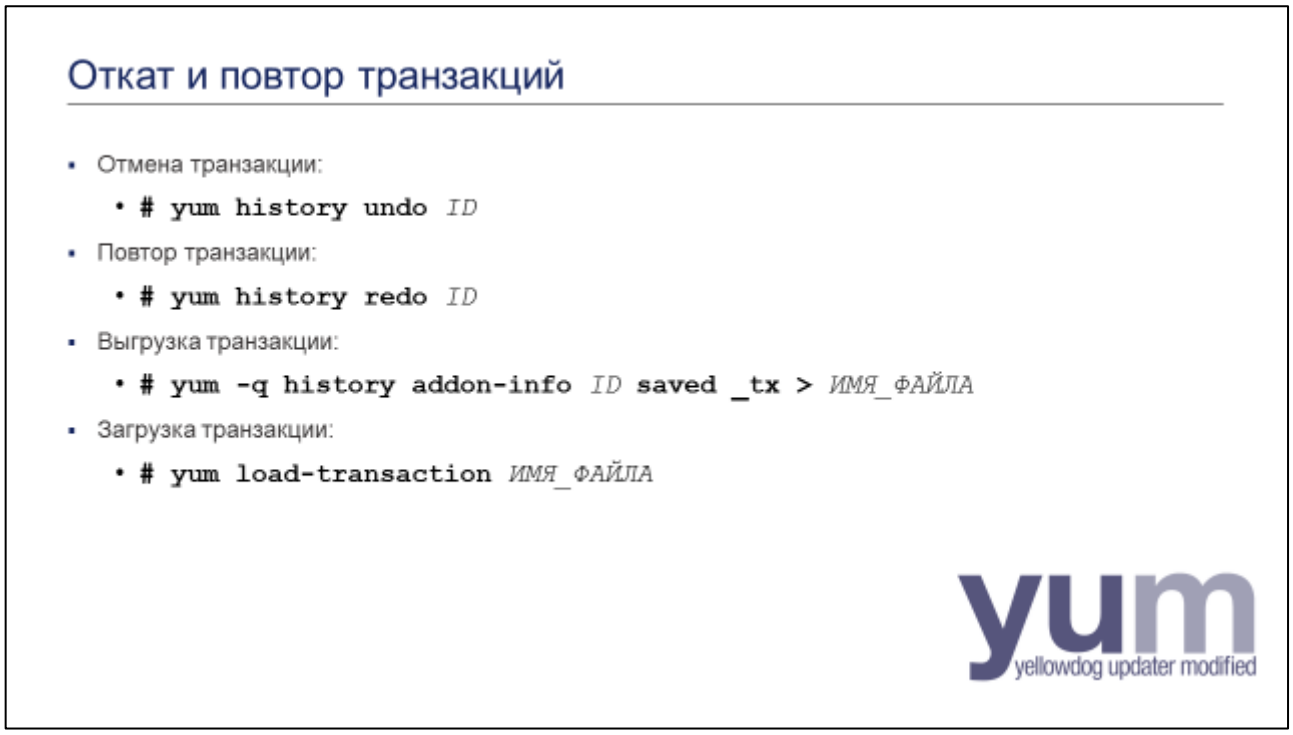
```
# yum history addon-info 4
Loaded plugins: fastestmirror, langpacks
Transaction ID: 4
Available additional history information:
  config-main
  config-repos
  saved_tx

history addon-info
```

В выводе команды **yum history addon-info** доступно три типа информации:

- **config-main** – Глобальные опции yum (располагаются в секции **[main]**, конфигурационного файла **/etc/yum.conf**), которая была использована в процессе выполнения транзакции.
- **config-repos** – Индивидуальные опции репозиториев **yum** (располагаются в секции **[repository]** конфигурационного файла репозитория.)
- **saved_tx** – Данные, которые могут быть использованы командой yum **load-transaction** в порядке повторения транзакции на другой машине.

Для отображения выбранного типа информации, выполните следующую команду от имени пользователя root:

```
# yum history addon-info id тип_информации
```



Независимо от просмотра истории транзакций, команда **yum history** предоставляет возможность отката и повторения выбранной транзакции. Для отката транзакции, воспользуйтесь следующей командой от имени пользователя root:

```
# yum history undo id
```

Для повторения выбранной транзакции, воспользуйтесь следующей командой:

```
# yum history redo id
```

Обе команды также принимают ключевое слово **last** для отката или повторения предыдущей транзакции.

Обратите внимание, что обе команды **yum history undo** и **yum history redo** откатывают или повторяют только те шаги, которые были выполнены в процессе выполнения транзакции. Если транзакция устанавливала пакет, то команда **yum history undo** удалит его, а если пакет был удален во время транзакции, то команда **yum history undo** установит его. Также эта команда попробует понизить версию всех обновленных пакетов до предыдущей версии, если более старая версия пакета по-прежнему доступна.

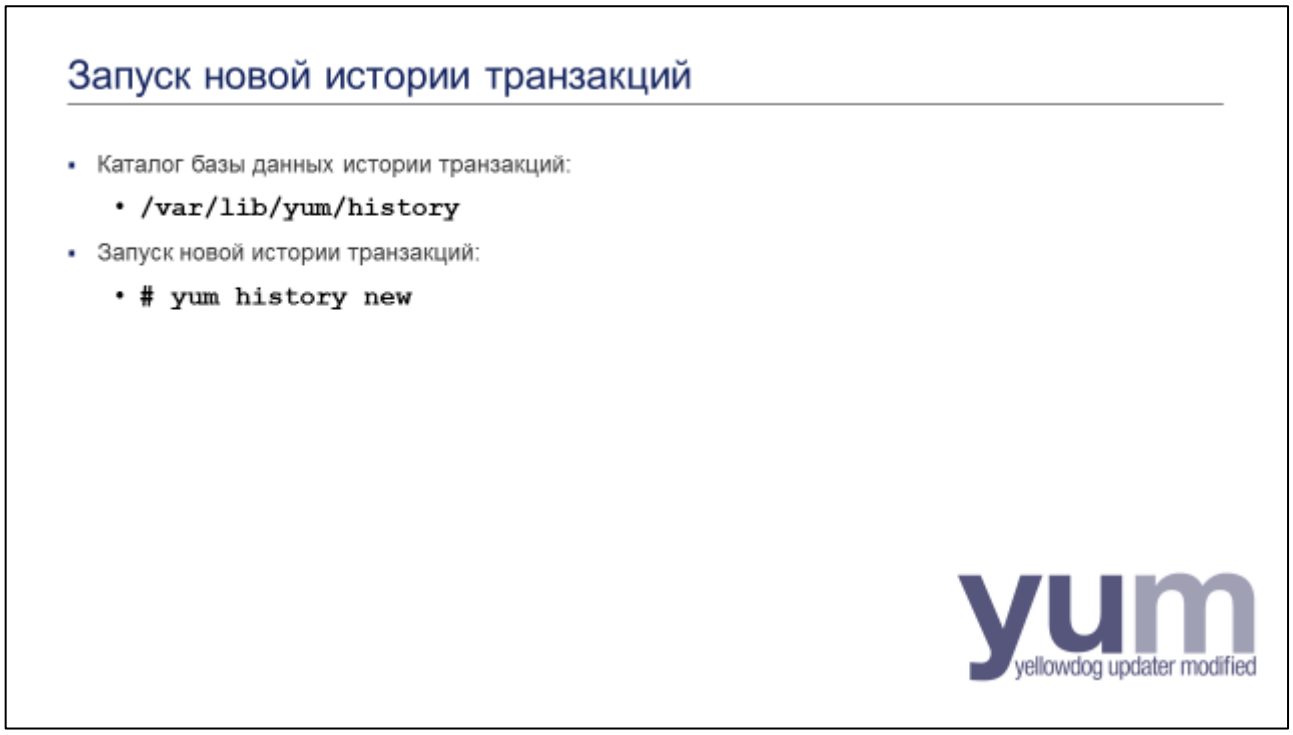
Когда требуется управлять несколькими одинаковыми системами, **yum** также позволяет выполнять транзакцию на одной из них, сохранять подробности транзакции в файле и после проверки, повторять эту транзакцию на других системах. Для сохранения подробностей транзакции в файле, выполните следующую команду от имени пользователя root:

```
# yum -q history addon-info id saved_tx > имя_файла
```

После того, как вы скопируете файл на целевую систему, вы можете повторить транзакцию при помощи следующей команды:

```
# yum load-transaction имя_файла
```

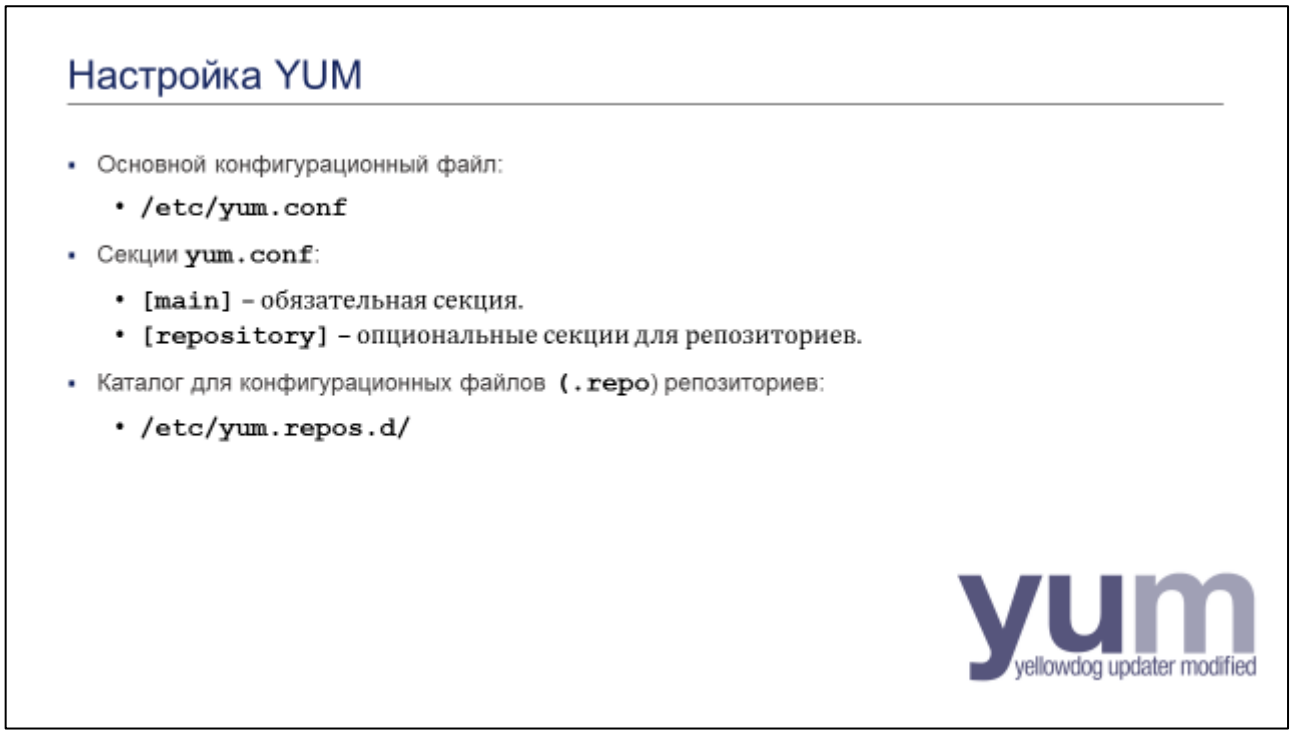
Вы можете настроить **load-transaction** на игнорирование недостающих пакетов или версию **rpmdb**. Дополнительную информацию об этом можно найти на странице руководства **yum.conf(5)**.



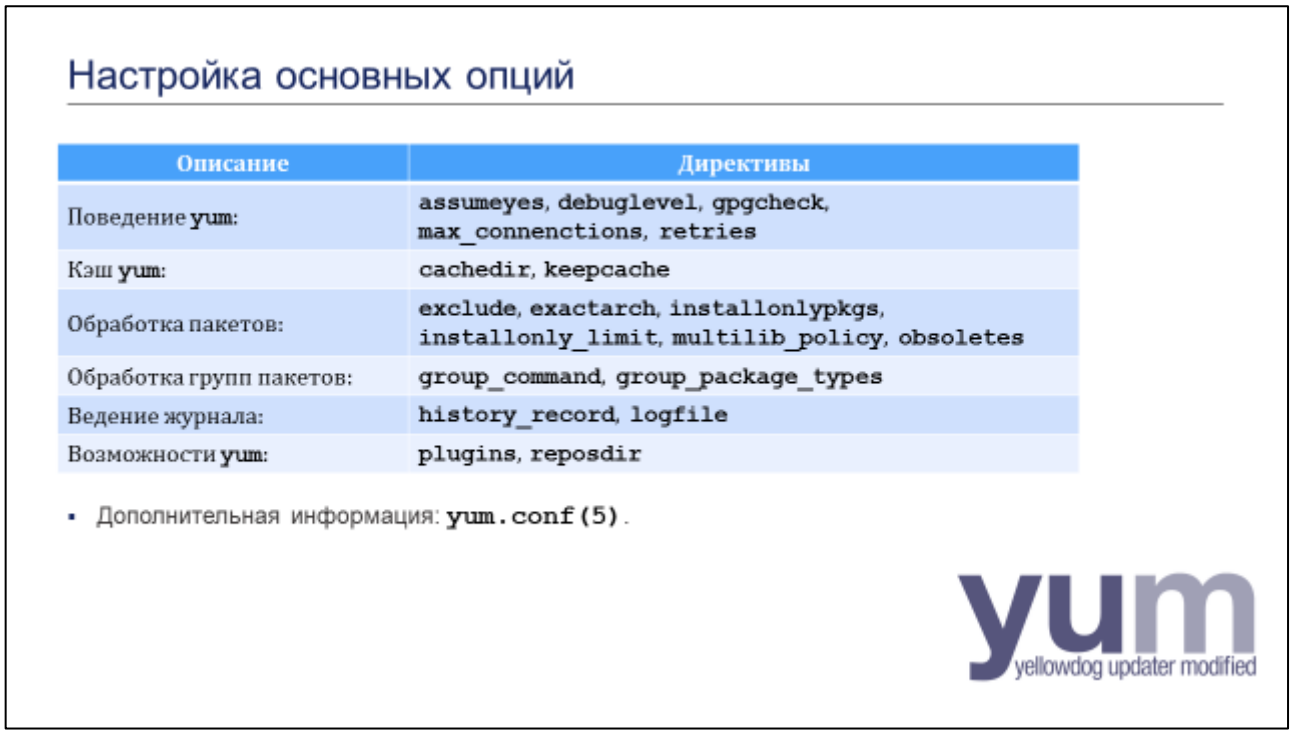
Yum хранит историю транзакций в отдельном файле базы данных SQLite. Для запуска новой истории транзакций, выполните следующую команду от имени пользователя root:

```
# yum history new
```

Это создаст новый, пустой файл базы данных в каталоге **/var/lib/yum/history**. Старый файл истории транзакции будет сохранен, но будет недоступен до тех пор, пока новый файл базы данных представлен в каталоге.



Информация по настройке для yum и связанных утилит располагается в **/etc/yum.conf**. Этот файл содержит одну обязательную секцию **[main]**, которая позволяет настраивать глобальные опции yum и может содержать одну или более секций **[repository]**, которые позволяют устанавливать специфичные опции для отдельных репозиториев. Однако, рекомендуется определять репозитории в новых или существующих файлах с расширением **.repo** в каталоге **/etc/yum.repos.d/**. Значения опций, определенные в секции **[repository]** переписывают значения, установленные в секции **[main]** файла **/etc/yum.conf**.



Конфигурационный файл **/etc/yum.conf** содержит только одну секцию **[main]**, которая состоит из пар ключей и значений, которые определяют, как будет работать yum. Вы можете добавить множество дополнительных опций в секцию **[main]**.

Пример конфигурационного файла **/etc/yum.conf**:

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
distroverpkg=centos-release

#  This is the default, …
```

Наиболее часто используемые опции в секции **[main]**:

assumeyes=*значение*

Опция assumeyes определяет запрашивать или не запрашивать подтверждение при критических действиях. Поддерживаемые значения:

- **0** (по умолчанию) – **yum** запрашивает подтверждение перед выполнением всех критических действий.
- **1** – **yum** не запрашивает подтверждение при критических действия также, как если бы команда yum выполнялась с опцией **-y** или **--assumeyes**.

cachedir=*каталог*

Данная опция устанавливает каталог, в котором **yum** сохраняет свой кэш и файлы базы данных. По умолчанию используется каталог **/var/cache/yum/$basearch/$releasever**.

`debuglevel=значение`

Данная опция указывает детализацию отладочного вывода, производимого yum. Значение может быть установлено в диапазоне от **1** до **10**. Установка более высокого значения **debuglevel** заставит yum отображать более подробный отладочный вывод. По умолчанию используется значение **2**, значение **0** отключит отображение отладочной информации.

`exactarch=значение`

Данная опция указывает определяет точно соответствие архитектуре при обновлении уже установленного пакета. Поддерживаемые значения:

- **0** – не соблюдает архитектуру при обновлении пакетов.
- **1** (по умолчанию) – не позволяет обновлять уже установленные пакеты, пакетами другой архитектуры. Например, нельзя установить более новый пакет с архитектурой 32 бита, если уже установлен пакет с архитектурой 64 бита.

`exclude=имя_пакета [имя_пакета]`

Опция exclude позволяет исключить пакеты, по ключевым словам, в процессе установки или обновления системы. Перечисление множества пакетов может быть заключено в кавычки и разделено пробелами. Также поддерживаются глобальные выражения (Glob Expressions) с подстановочными знаками (***** и **?**).

`gpgcheck=значение`

Опция gpgcheck заставляет yum выполнять проверку подписи GPG на пакетах. Поддерживаемые значения:

- **0** – Отключает проверку GPG подписей на пакетах во всех репозиториях, в том числе при локальной установке пакетов.
- **1** (по умолчанию) – Включает проверку подписей GPG на всех пакетах во всех репозиториях, в том числе при локальной установке.

`group_command=значение`

Данная опция указывает поведение команд **yum group install**, **yum group upgrade** и **yum group remove** при обработке группы пакетов. Поддерживаемые значения:

- **simple** – Устанавливать всех членов группы пакетов. Обновлять только ранее установленные пакеты, но не устанавливать пакеты, которые были добавлены в группу позже.
- **compat** – Похоже на **simple**, но **yum upgrade** также устанавливает пакеты, которые были добавлены в группу с момента предыдущего обновления.
- **objects** (по умолчанию) – **yum** продолжает отслеживать ранее установленные группы и пакеты, установленные в группе и отдельно.

`group_package_types=package_type [more_package_types]`

Указывает какой тип пакета (**optinal**, **default** или **mandatory**) устанавливается, когда вызывается команда yum group install. По умолчанию используется тип пакета **mandatory**.

`history_record=значение`

Включает или отключает ведение истории транзакций.

- **0** – yum не записывает историю транзакций.

- **1** (по умолчанию) – **yum** записывает историю транзакций. Эта операция требует дополнительное место на диске и дополнительное время в транзакции, но предоставляет много информации о прошедших операциях, которая может быть отображена при помощи команды **yum history**.

ПРИМЕЧАНИЕ

Yum использует запись истории для определения изменений в базе данных **rpmdb**, которые были внесены без участия **yum**. В таких случаях yum отображает предупреждение и автоматически ищет проблему, вызванную изменением **rpmdb**. Когда опция **history_record** отключена, **yum** не может определить изменения и не выполняет автоматическую проверку.

`installonlypkgs=`*`разделенный пробелами список пакетов`*

Указывает список пакетов, разделенных пробелами, которые **yum** может установить, но не может обновить.

`installonly_limit=`*`значение`*

Указывает как много пакетов, указанных в директиве installonlypkgs может быть установлено одновременно.

`keepcache=`*`значение`*

Определяет удержание заголовков и пакетов после успешной установки в кэше.

Значение может принимать:

- **0** (по умолчанию) – не хранить заголовки и пакеты кэше после успешной установки.
- **1** – хранить заголовки и пакеты кэше после успешной установки.

`logfile=`*`имя_файла`*

Определяет расположение вывода журнала, определяется указанием абсолютного пути. По умолчанию используется: **/var/log/yum.log**.

`max_connenctions=`*`число`*

Определяет максимальное число конкурентных подключений. По умолчанию **5**.

`multilib_policy=`*`значение`*

Определяет поведение установки, в случае доступности нескольких архитектур.

Значение может принимать:

- **best** – устанавливать пакеты, соответствующие архитектуре системы в первую очередь. Например, для систем на базе AMD64 всегда будет устанавливаться 64-битная версия пакета.
- **all** – устанавливать пакеты всех доступных архитектур. Например, для систем на базе AMD64 всегда будет устанавливаться обе версии пакета, если доступны i686 и AMD64.

`obsoletes=`*`значение`*

Определяет обновления пакетов до более новой версии. Когда один пакете определяет устаревание другого пакета, который при установке будет удален.

Значение может принимать:

- **0** – отключает логику обработки устаревания yum в процессе выполнения обновления.
- **1** (по умолчанию) – включает логику обработки устаревания yum в процессе выполнения обновления.

```
plugins=значение
```

Данный глобальный параметр включает или отключает плагины yum.

Значение может принимать:

- **0** – Глобально отключает все плагины yum.
- **1** (по умолчанию) - Глобально включить все плагины yum, при этом остается возможность отключить необходимые плагины выставив параметр **enabled=0** в конфигурационном файле плагина.

ПРИМЕЧАНИЕ

Не рекомендуется отключать все плагины, так как некоторые плагины предоставляют важные службы **yum**. Глобальное отключение как правило используется в целях диагностики и выявления неисправностей.

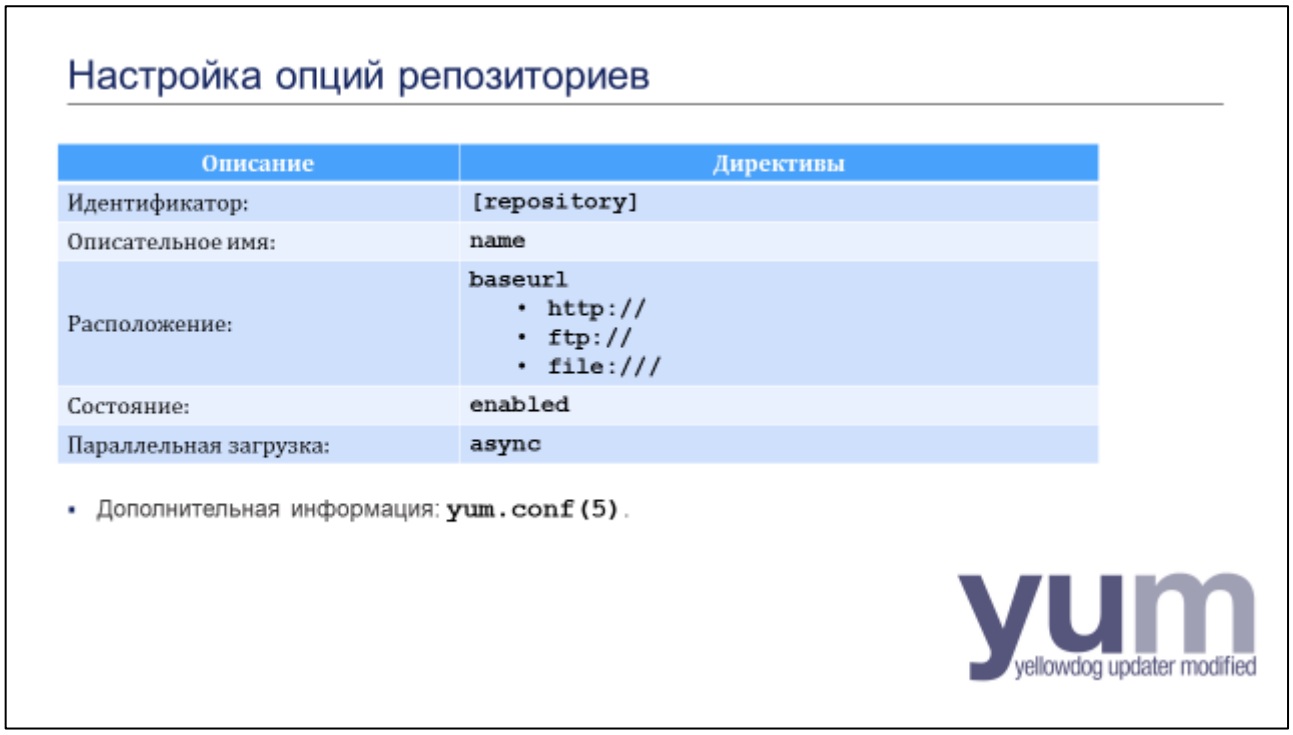
```
reposdir=каталог
```

Указывает абсолютный путь к каталогу с конфигурационными файлами репозиториев (**.repo**). Все **.repo** файлы содержат информацию о репозиториях (подобно секции **[repository]** файла **/etc/yum.conf**). Yum собирает информацию из всех файлов **.repo** и секции **[repository]** файла **/etc/yum.conf** и создает список репозиториев, который используется в транзакциях. Если опция **reposdir** не указана используется каталог по умолчанию **/etc/yum.repos.d/**.

```
retries=значение
```

Указывает число попыток получить доступ к файлу перед выдачей ошибки. Значение должно быть целочисленным, указанным в диапазоне от **0** и больше. Значение **0** заставляет делать бесконечно число попыток. Значение по умолчанию – **10**.

Для получения полного списка всех доступных опций **[main]**, влспльзуйтесь секцией **[main] OPTIONS** страницы руководства **yum.conf(5)**.



Секция **[repository]**, в которой **repository** – это уникальный идентификатор, такой как **redos_repo** (пробелы не разрешены), позволяет вам определить индивидуальные репозитории yum. Во избежание конфликтов, настраиваемые репозитории не должны использовать имена репозиторриев RedOS.

Следующий пример демонстрирует минимальную форму определения секции **[repository]**:

```
[repository]
name=имя_репозитория
baseurl=url_репозитория
```

Каждая секция **[***repository***]** должна содержать следующие директивы:

```
name=имя_репозитория
```

Указывает описательное имя для репозитория.

```
baseurl=url_репозитория
```

Указывает URL до каталога, содержащего репозиторий:

- URL для репозитория доступного по протоколу HTTP: `http://путь/до/репозитория`
- URL для репозитория доступного по протоколу FTP: `ftp://путь/до/репозитория`
- URL для репозитория на локальном диске: `file:///путь/до/репозитория`

Если указанный онлайн репозиторий требует базовую аутентификацию HTTP, вы можете указать имя пользователя и пароль в строке URL, в следующем формате:

```
имя_пользователя:пароль@ссылка.
```

Например: `http://user:password@www.example.com/repo/`.

Обычно URL – это HTTP-ссылка, следующего вида:

baseurl=http://*сервер***/***releases***/$releasever/***server***/$basearch/os/**

Где **$releasever** и **$basearch** – это переменные.

Прочие полезные директивы секции **[repository]**:

```
enabled=значение
```

Определяет использование репозитрия (включен/отключен).

Значение может принимать:

- **0** – не использовать репозиторий в качестве источника пакетов.
- **1** - использовать репозиторий в качестве источника пакетов.

Включать и откалючать репозитории можно при помощи опций команды yum:

--enablerepo=*имя_репозитория* и **--disablerepo=***имя_репозитория*.

```
async=значение
```

Управляет параллельной загрузкой пакетов из репозитория.

Значение может принимать:

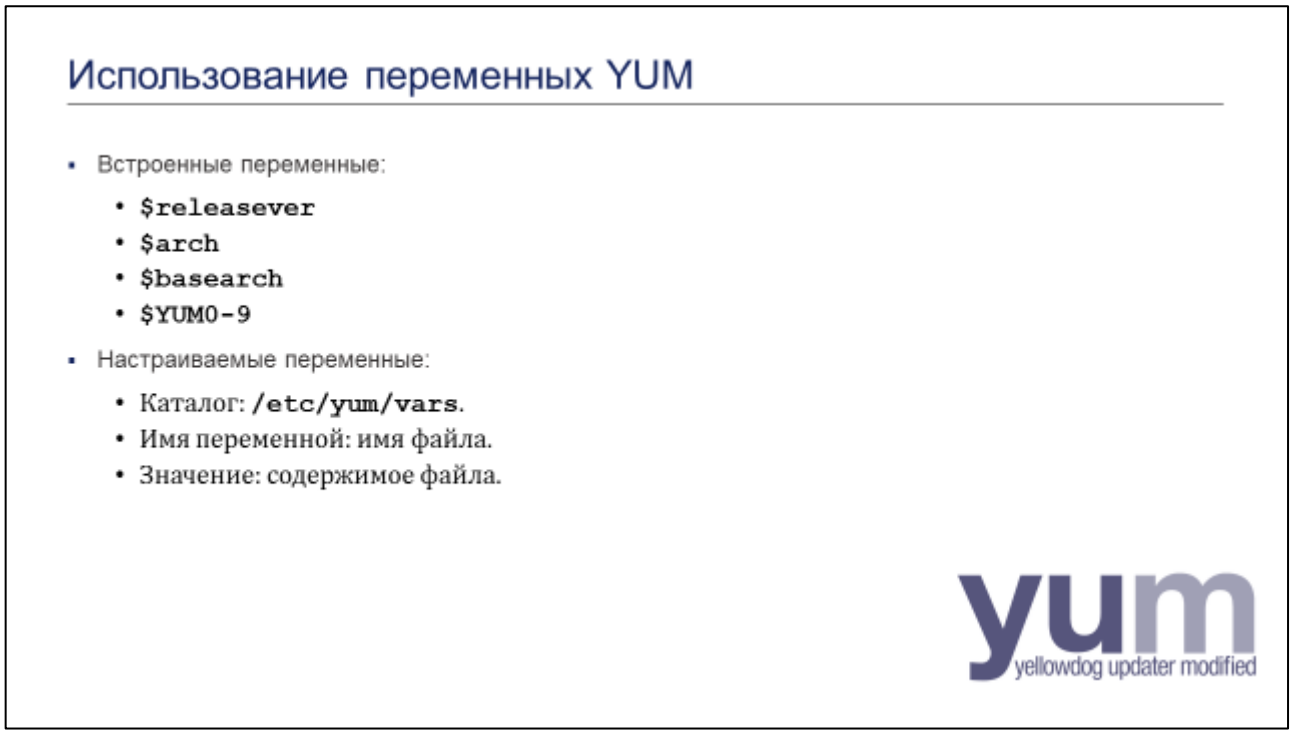
- **auto** (по умолчанию) – если доступна, используется параллельная загрузка. Yum автоматически отключает параллельную загрузку для репозиториев созданных при помощи плагинов.
- **on** – включение параллельной загрузки.
- **off** – отключение параллельной загрузки.

Существует множество других опций для секции **[repository]**, часть из них полностью повторяет опции секции **[main]**. Для получения полного списка доступных опций, обратитесь к секции **[repository] OPTIONS** страницы руководства **yum.conf(5)**.

Пример конфигурационного файла **/etc/yum.repos.d/RedOS-Base.repo**:

```
# RedOS-Base.repo
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#

[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=o
s&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=u
pdates&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

```
#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=e
xtras&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=c
entosplus&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$basearch/
gpgcheck=1
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```



Вы можете использовать и ссылаться на следующие встроенные переменные в командах yum и всех конфигурационных файлах:

`$releasever`

Вы можете использовать эту переменную для ссылки на версию релиза CentOS. Yum применяет значение **$releasever** из строки **distroverpkg=***value* конфигурационного файла **/etc/yum.conf**. Если строка не определена в файле **/etc/yum.conf**, в таком случае yum выбирает номер версии из пакета centos-release-версия, который предоставляет файл **/etc/redhat-release**.

`$arch`

Вы можете использовать эту переменную для ссылки на архитектуру процессора в том же виде как возвращает функция Python **os.uname()**. Корректные значения для **$arch** включают **i586**, **i686** и **x86_64**.

`$basearch`

Вы можете использовать эту переменную для ссылки на базовую архитектуру системы. Например, архитектуры **i586** и **i686** базируются на архитектуре **i386**, а AMD64 и Intel64 базируются на архитектуре **x86_64**.

`$YUM0-9`

Эти 10 переменных заменяются значениями переменных окружения оболочки с теми же именами. Если переменная определена в конфигурационном файле (например, в **/etc/yum.conf**) и переменная отсутствует в окружении оболочки, в таком случае переменная конфигурационного файла не заменяется.

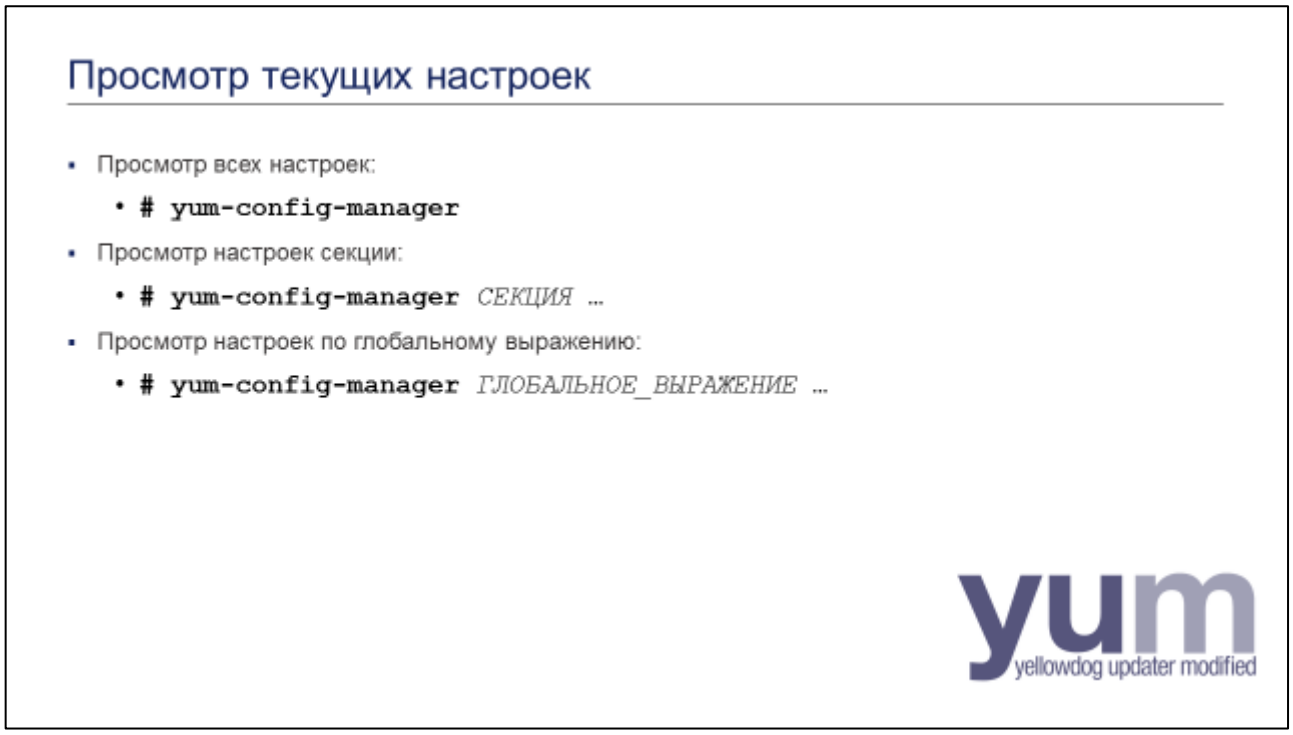
Для определения настраиваемой переменной или перезаписи существующей создайте файл с таким де именем (без знака **$**) в каталоге **/etc/yum/vars** и добавьте в него необходимое значение в первую строку.

Например, описание репозитория часто использует имя операционной системы. Для определения новой переменной $osname, создайте новый файл **/etc/yum/vars/osname** со значением «**Red Operating System**» в первой строке:

```
# echo "Red Operating System" > /etc/yum/vars/osname
```

Теперь вы можете использовать ее в файлах **.repo**:

```
name=$osname 7.2
```



Для отображения текущих значений глобальных опций yum (опции, указанные в секции **[main]** конфигурационного файла **/etc/yum.conf**) выполните команду **yum-config-manager**:

```
# yum-config-manager
```

Для отображения информации из другой конфигурационной секции или секций, используйте команду в следующем виде:

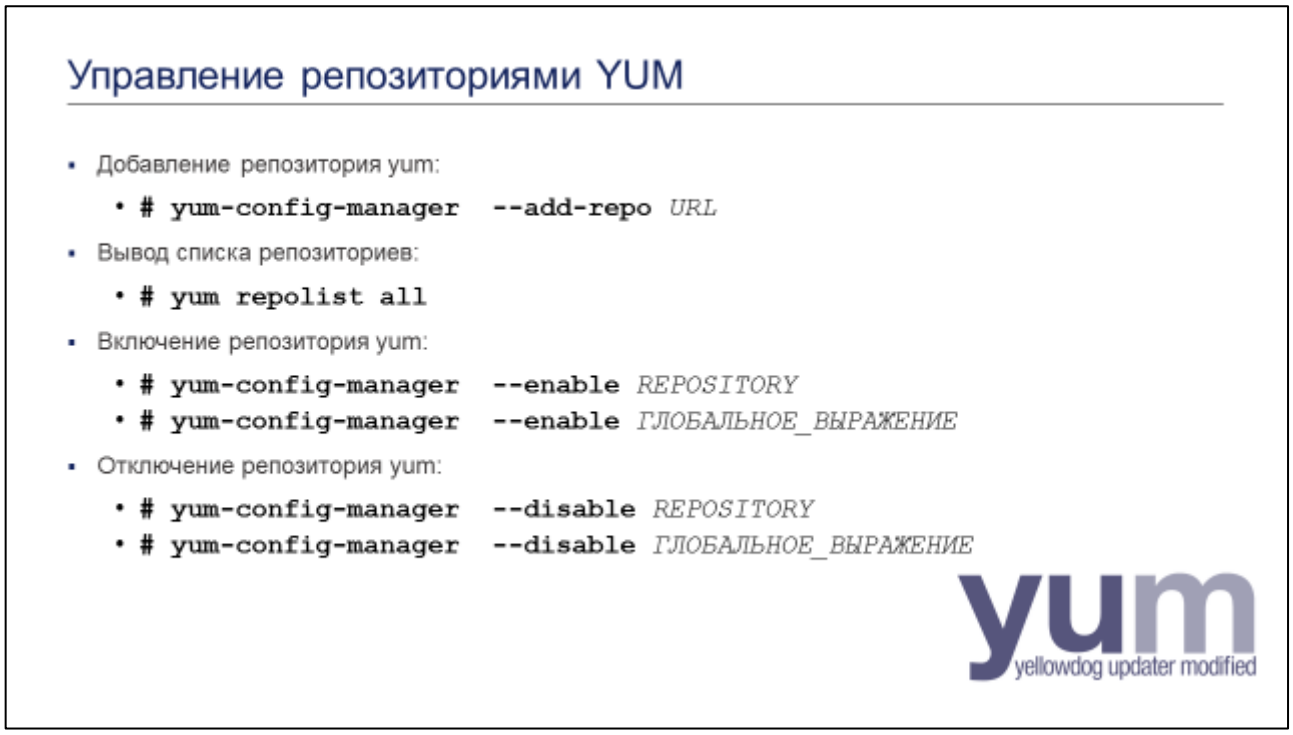
```
# yum-config-mamager секция…
```

Также можно использовать глобальные выражения (Glob Expression):

```
# yum-config-mamager глобальное_выражение…
```

Пример вывода настроек секции **[main]** конфигурационного файла **/etc/yum.conf**:

```
# yum-config-manager main
Loaded plugins: fastestmirror, langpacks
================================== main ==================================
[main]
alwaysprompt = True
assumeno = False
assumeyes = False
autocheck_running_kernel = True
bandwidth = 0
cache = 0
cachedir = /var/cache/yum/x86_64/7
check_config_file_age = True
clean_requirements_on_remove = False
color = auto
color_list_available_downgrade = dim,cyan
color_list_available_install = normal
color_list_available_reinstall = bold,underline,green
color_list_available_running_kernel = bold,underline
color_list_available_upgrade = bold,blue
…
```



Для определения нового репозитория **yum**, вы можете добавить секцию **[repository]** в файл **/etc/yum.conf** или или файл **.repo** в каталог **/etc/yum.repos.d/**. Все файлы с расширением **.repo** в этом каталоге читаются yum, и это рекомендованный способ определения репозитория.

Как правило, репозитории yum предоставляют собственные файл **.repo**. Для добавления репозитория в систему и включения его используйте следующую команду:

```
# yum-config-manager --add-repo URL
```

Пример добавления репозитория:

```
# yum-config-manager --add-repo http://172.20.70.251/repo/redos72
Loaded plugins: fastestmirror, langpacks
adding repo from: http://172.20.70.251/repo/redos72

[172.20.70.251_repo_redos72]
name=added from: http://172.20.70.251/repo/redos72
baseurl=http://172.20.70.251/repo/redos72
enabled=1
```

Для вы вода полного списка репозиториев воспользуйтесь командой:

```
# yum repolist all
```

Для включения репозитория **yum**, воспользуйтесь следующей командой:

```
# yum-config-manager --enable repository…
```

Также можно использовать глобальные выражения (Glob Expression):

```
# yum-config-mamager --enable глобальное_выражение…
```

Пример включения репозитория:

```
# yum-config-manager --enable me\*
Loaded plugins: fastestmirror, langpacks
================================ repo: c7-media ================================
[media]
```

```
async = True
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7
baseurl = file:///media/RedOS/,
   file:///media/cdrom/,
   file:///media/cdrecorder/
cache = 0
cachedir = /var/cache/yum/x86_64/7/media
check_config_file_age = True
cost = 1000
deltarpm_metadata_percentage = 100
deltarpm_percentage =
enabled = 1
…
```

После успешного включения репозитория, команда **yum-config-manager** отображает его текущие параметры.

Для отключения репозитория yum, воспользуйтесь следующей командой:

```
# yum-config-manager --disable repository…
```

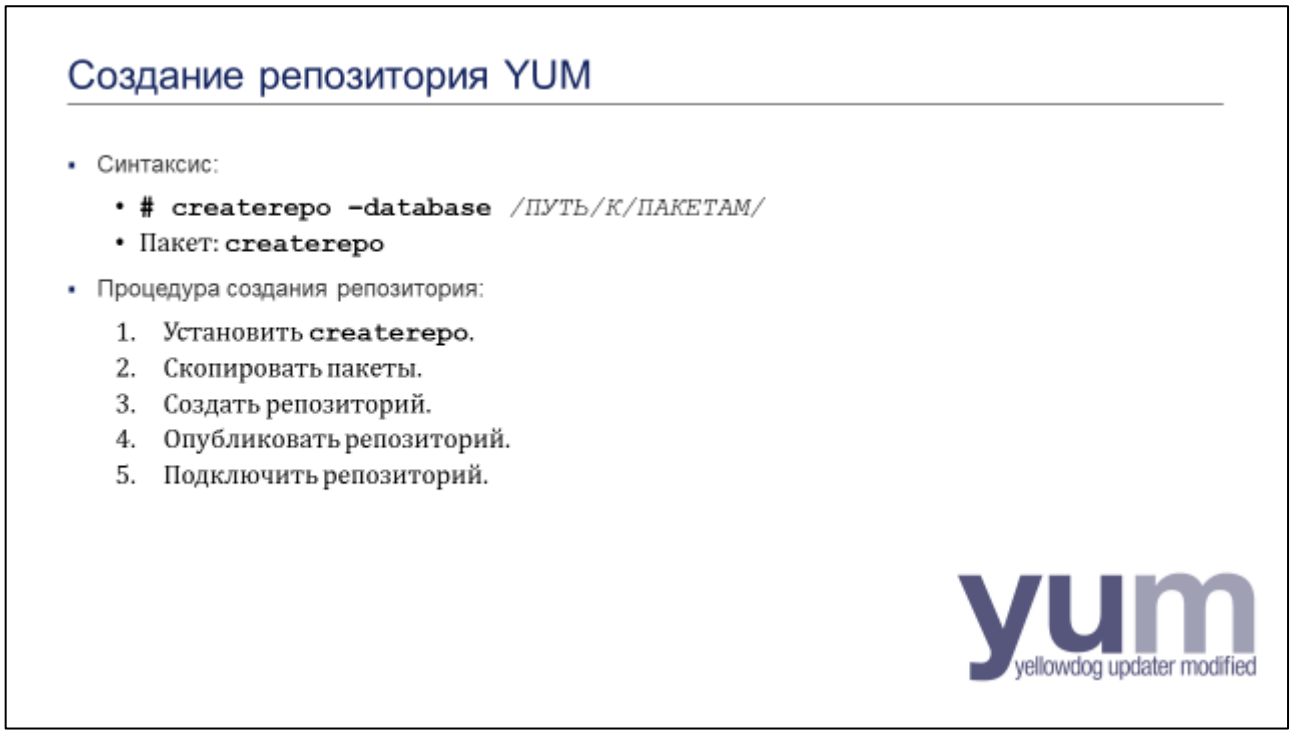
Также можно использовать глобальные выражения (Glob Expression):

```
# yum-config-mamager --disable глобальное_выражение…
```

Пример отключения репозитория:

```
# yum-config-manager --disable me\*
Loaded plugins: fastestmirror, langpacks
================================ repo: c7-media ================================
[media]
async = True
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7
baseurl = file:///media/RedOS/,
   file:///media/cdrom/,
   file:///media/cdrecorder/
cache = 0
cachedir = /var/cache/yum/x86_64/7/c7-media
check_config_file_age = True
cost = 1000
deltarpm_metadata_percentage = 100
deltarpm_percentage =
enabled = 0
…
```

После успешного отключения репозитория, команда **yum-config-manager** отображает его текущие параметры.



Для настройки репозитория yum воспользуйтесь следующими командами:

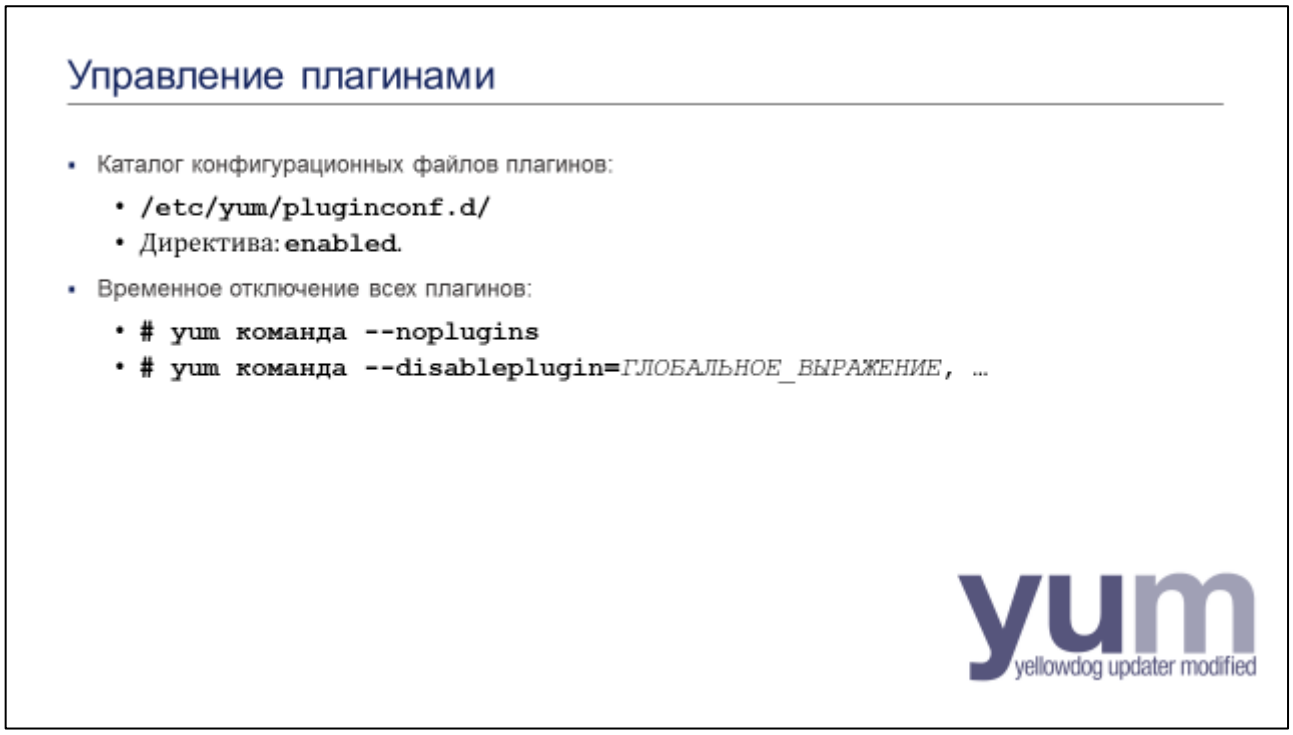
1. Установите пакет **createrepo**:

```
# yum install createrepo
```

2. Скопируйте все необходимые пакеты в единое расположение, например **/mnt/lrepo**.
3. Создайте репозиторий в каталоге:

```
# createrepo --database /mnt/lrepo
```

Это создаст все необходимые файлы в каталоге, в том числе базу данных **sqlite** для ускорения операций **yum**.



Yum предоставляет плагины, которые расширяют и улучшают его возможности. Набор необходимых плагинов устанавливается по умолчанию. Yum всегда информирует о том, какие плагины используются во время любой операции, например:

```
# yum info yum
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.nsu.ru
```

Обратите внимание, что имена, указанные после слов **Loaded plugins**, это имена, которые можно использовать в командах, например **--disableplugin=***plugin_name*.

Для того чтобы включить плагины, убедитесь, что в строке **plugins=** в секции **[main]** конфигурационного файла **/etc/yum.conf** установлено значение **1**.

```
plugins=1
```

Для отключения всех плагинов выставьте значение **plugins=0**.

ПРИМЕЧАНИЕ

Не рекомендуется отключать все плагины, так как некоторые плагины предоставляют важные службы **yum**. Глобальное отключение как правило используется в целях диагностики и выявления неисправностей.

Для каждого установленного пллагина имеется свой собственный конфигурационный файл в каталоге **/etc/yum/pluginconf.d/**. Вы можете установить специальные параметры плагинов в этих файлах. Пример конфигурационного файла плагина **aliases – aliases.conf**:

```
[main]
enabled=1
```

Подобно конфигурационному файлу **/etc/yum.conf**, конфигурационные файлы плагинов содержит секцию **[main]**, в которой присутствует опция **enabled=**, позволяющая отключить плагин. Если данная опция отсутствует в файле вы можете добавить ее вручную.

Если плагины отключены глобально, в секции **[main]** конфигурационного файла **/etc/yum.conf**, значение опции **enabled=** в конфигурационных файлах плагинов игнорируется.

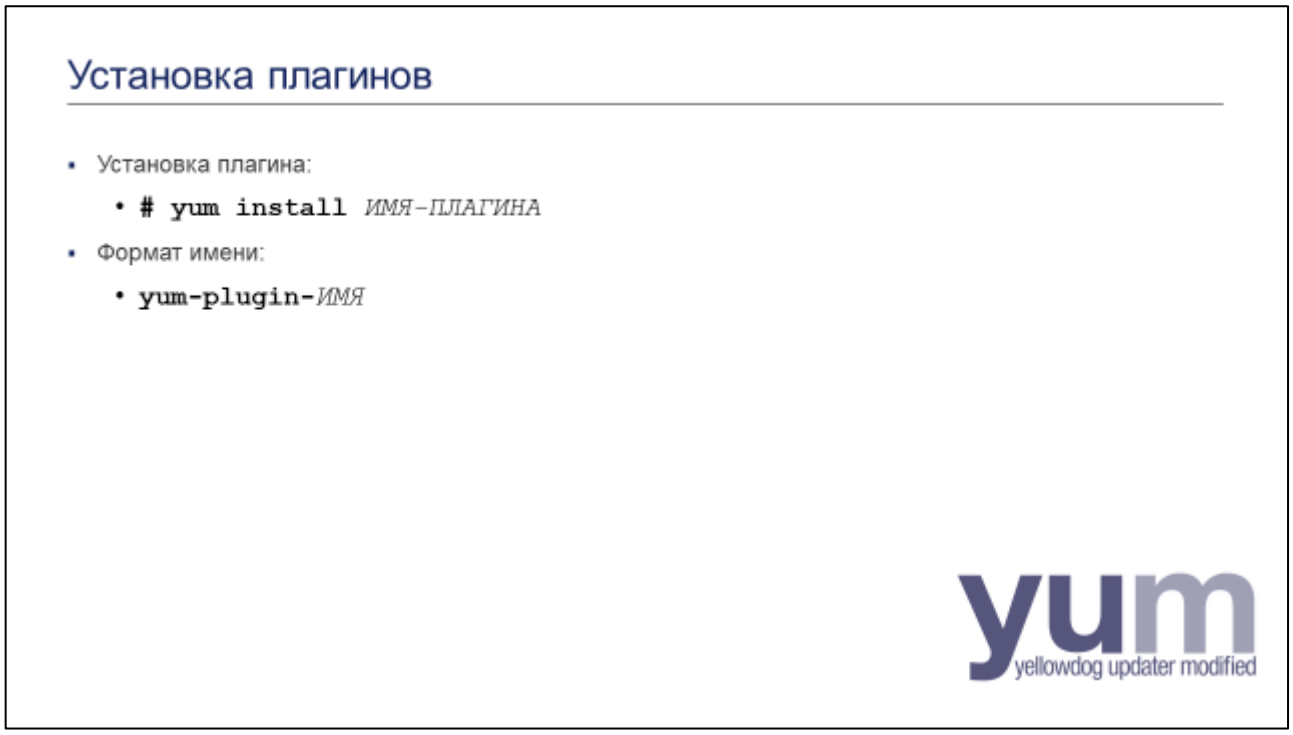
Для разового отключения всех плагинов можно использовать опцию **--noplugins** команды **yum**.

Для разового отключения одного или нескольких плагинов воспользуйтесь опцией **--disableplugin** команды **yum**. Например, для отключения плагина aliases, в процессе обновления, воспользуйтесь следующей командой:

```
# yum update --disableplugin=aliases
```

Для указания нескольких плагинов, их нужно указывать через запятую, также поддерживаются глобальные выражения (Glob Expressions):

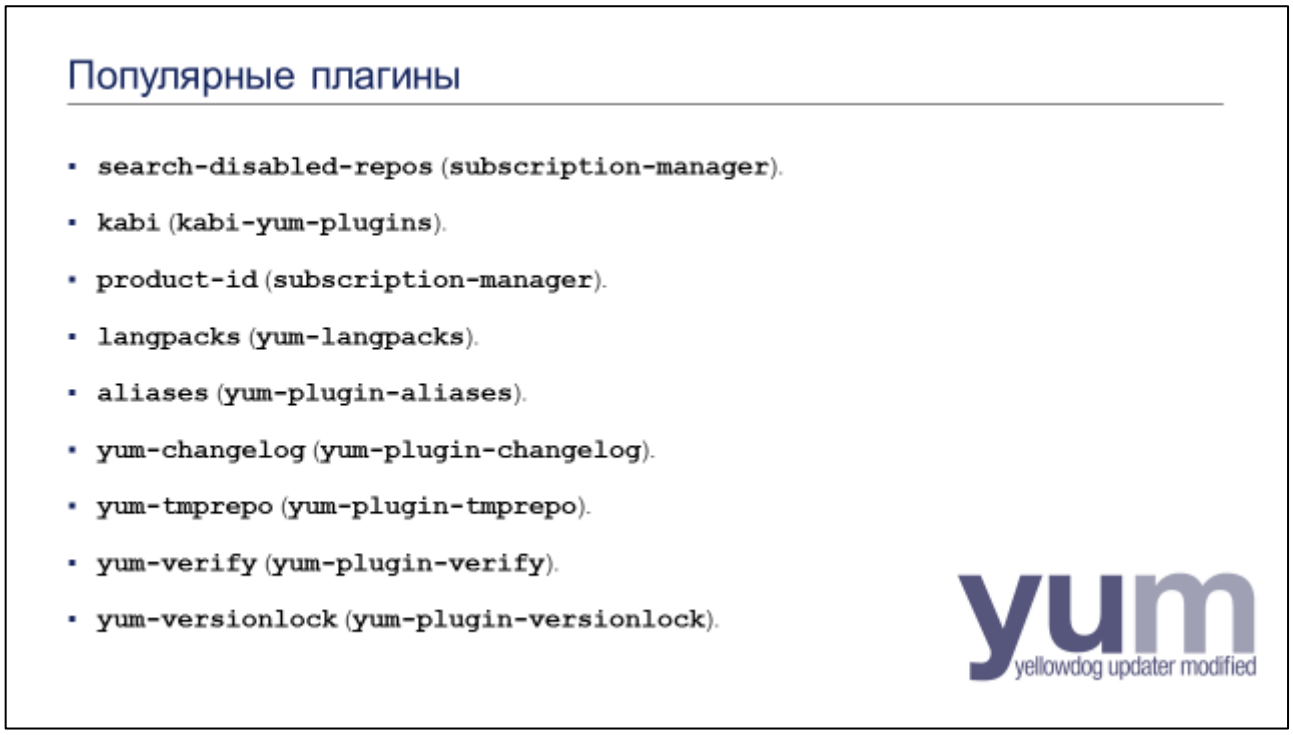
```
# yum update --disableplugin=aliases, lang*
```



Плагины yum распространяются в пакетах, соответствующих следующему соглашению именования: **yum-plugin-***имя-плагина*, но не всегда. Например, пакет предоставляющий плагин **kabi** называется **kabi-yum-plugins**.

Пример установки плагина **aliases**:

```
# yum install yum-plugin-aliases
```



Следующий список предоставляет описание и инструкции по использованию популярных плагинов yum. В скобках указаны имена пакетов, предоставляющих плагин.

`search-disabled-repos (subscription-manager)`

Плагин **`search-disabled-repos`** позволяет временно или на постоянной основе включить отключенный репозиторий, чтобы помочь с разрешением зависимостей. Когда этот плагин включен и yum не может завершить установку из-за неразрешенных зависимостей, плагин временно включает отключенные репозитории и повторно пробует выполнить операцию. Если установка проходит успешно, yum предлагает оставить включенным использованный репозиторий. Обратите внимание, что плагин работает только с репозиториями, управляемыми через **`subscription-manager`** и не работает с настраиваемыми репозиторриями.

ПРИМЕЧАНИЕ

Если yum выполняется с опциями **`--assumeyes`** или **`-y`**, или директива **`assumeyes`** включена в **`/etc/yum.conf`** плагин включает репозитории временно и на постоянной основе без запросов подтверждения. Это может привести к проблемам, связанным с включением репозиториев, которые вы не хотите использовать.

Поддерживаемые директивы конфигурационного файла **`search-disabled-repos.conf`**:

Директивы	Описание
`enabled=`*`значение`*	Позволяет включать или отключать плагин, поддерживаемые значения: **`1`** (включен) и **`0`** (отключен). По умолчанию включен.
`notify_only=`*`значение`*	Позволяет ограничить функционал только до уведомлений, поддерживаемые значения: **`1`** (только уведомления) и **`0`** (позволяет управлять поведением yum). По умолчанию только уведомляет.

ignored_repos=*репозитории*	Позволяет указать репозитории, которые не будут включены плагином.

kabi (kabi-yum-plugins)

Плагин **kabi** проверяет пакеты обновления драйверов на соответствие официальному прикладному бинарному интерфейсу ядра (Kernel Application Binary Interface, KABI). При включенном плагине, когда пользователь попытается установить пакет, который использует символы ядра, не представленные в списке разрешенных, сообщение с предупреждением будет записано в системный журнал. Дополнительно, если настроить плагин на запуск в принудительном режиме это не позволит таким пакетам устанавливаться.

Для настройки плагина **kabi**, отредактируйте конфигурационный файл, расположенный в файле **/etc/yum/pluginconf.d/kabi.conf**. Список директив, которые можно использовать в секции **[main]** представлен в таблице:

Директивы	Описание
enabled=*значение*	Позволяет включать или отключать плагин, поддерживаемые значения: **1** (включен) и **0** (отключен). Когда установлен, по умолчанию включен.
whitelists=*каталог*	Позволяет указать каталог, который содержит файлы с поддерживаемыми символами ядра. По умолчанию, плагин **kabi** использует файлы, предоставляемые пакетом **kernel-abi-whitelists** (каталог **/usr/lib/modules/kabi-rhel70/**).
enforce=*значение*	Позволяет включить или отключить принудительный режим, поддерживаемые значения: **1** (включен) и **0** (отключен). По умолчанию данная опция закомментирована и плагин **kabi** отображает только предупреждающее сообщение.

product-id (subscription-manager)

Плагин **product-id** управляет сертификатами идентификации продуктов, устанавливаемыми из Content Delivery Network. Плагин **product-id** установлен по умолчанию.

langpacks (yum-langpacks)

Плагин **langpacks** используется для поиска языковых пакетов выбранных языков для каждого устанавливаемого пакета. Плагин **langpacks** установлен по умолчанию.

aliases (yum-plugin-aliases)

Плагин **aliases** добавляет опцию командной строки **alias**, которая позволяет настройку и использование псевдонимов для команд **yum**.

yum-changelog (yum-plugin-changelog)

Плагин **aliases** добавляет опцию командной строки **--changelog**, которая позволяет просматривать журнал изменения пакета до и после обновления.

yum-tmprepo (yum-plugin-tmprepo)

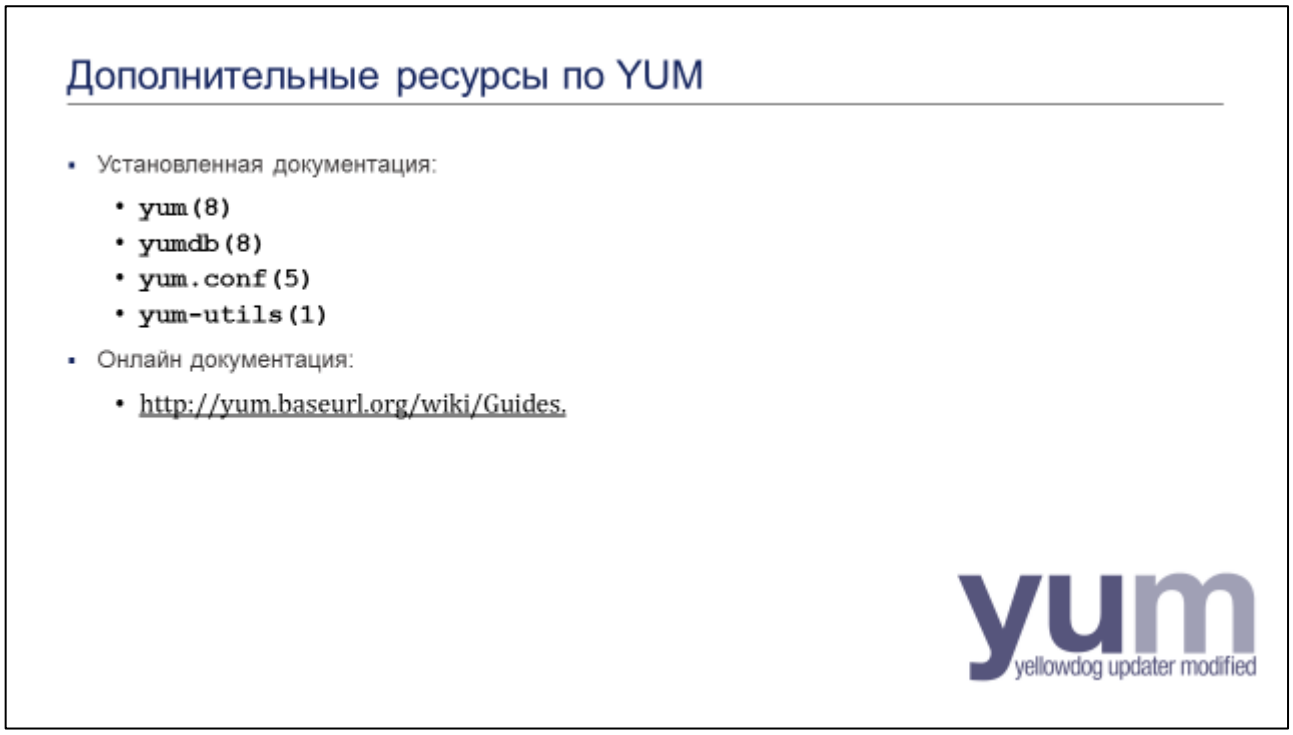
Плагин **yum-tmprepo** добавляет опцию командной строки **--tmprepo**, которая берет URL файла репозитория, загружает и включает его только на одну транзакцию. Данный плагин пытается обеспечить безопасное временное использование репозитория. По умолчанию не позволяет отключать проверку GPG.

yum-verify (yum-plugin-verify)

Плагин yum-verify добавляет опции командной строки **verify**, **verify-rpm** и **verfy-all** для просмотра данных проверки на системе.

yum-versionlock (yum-plugin-versionlock)

Плагин **yum-versionlock** исключает прочие версии выбранного пакета, что позволяет защитить пакеты от обновления новыми версиями. При помощи опции командной строки **versionlock** можно просматривать и редактировать список заблокированных пакетов.



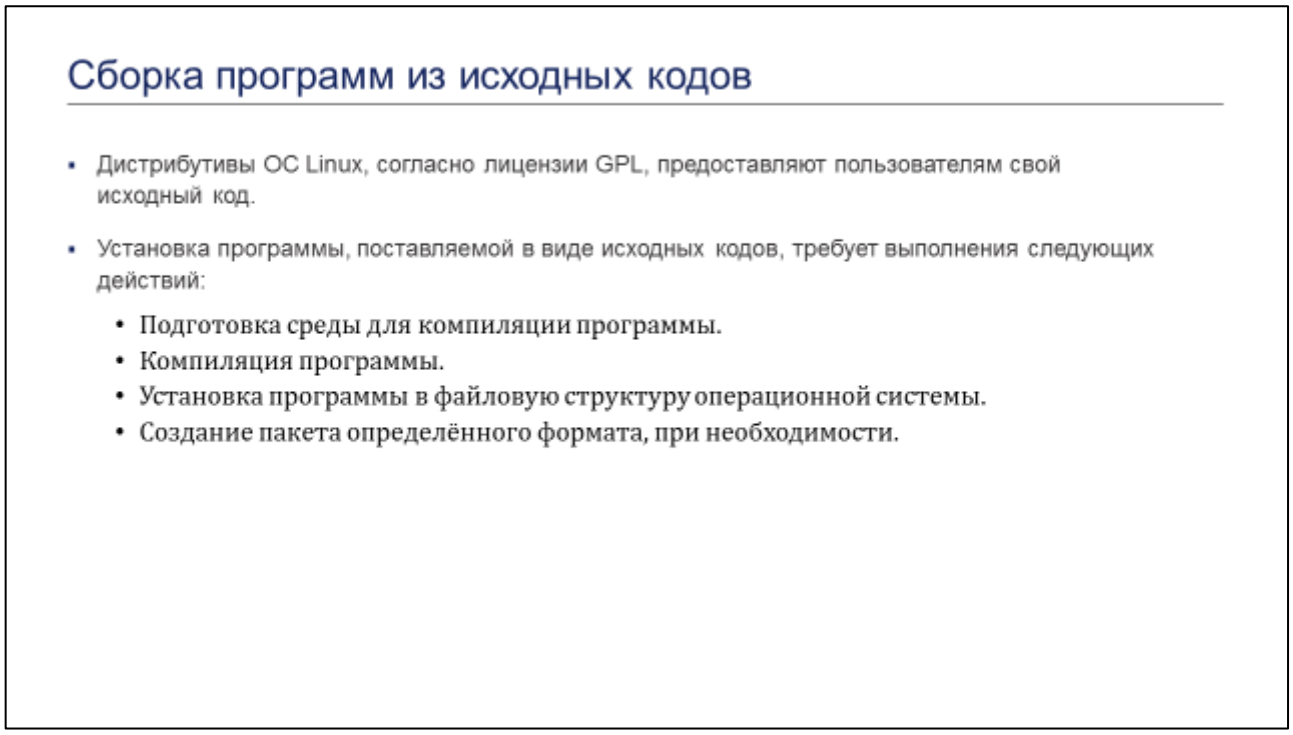
Для получения дополнительной информации по управлению пакетами в CentOS (RHEL) обратитесь к следующим ресурсам.

Установленная документация:

- **yum(8)** – страница руководства утилиты командной строки **yum**, предоставляет полный список поддерживаемых опций и команд.
- **yumdb(8)** – страница руководства утилиты командной строки **yumdb**, описывает использование утилиты для запросов и, при необходимости, изменения базы данных.
- **yum.conf(5)** – страница руководства конфигурационного файла **yum.conf**, содержащая все доступные опции настройки **yum**.
- **yum-utils(1)** – страница руководства, содержащая краткое описание дополнительных утилит управления настройками **yum**, управления репозиториями и работы с базой данных **yum**.

Онлайн документация:

- Руководства yum (http://yum.baseurl.org/wiki/Guides).



Операционная система UNIX появилась на свет одновременно с языком программирования C, более того, язык C был разработан для создания ОС UNIX. В последнее время, всё чаще используется и C++.

Лицензия GPL обязывает разработчиков предоставлять исходные коды своих программных продуктов, а соответственно, используя операционную систему Linux, рано или поздно придётся столкнуться с установкой программы, поставляемой в виде исходных кодов.

Для выполнения процедуры сборки программы используется программа gcc (GNU Compiler Collection). На сегодняшний день, GCC объединяет компиляторы для различных языков программирования: C, C++, Fortran, Ada, Java и т.д. GCC состоит из 4-х основных компонент:

- Препроцессор модифицирует код программы для подготовки к компиляции.
- Компилятор обрабатывает исходный код программы и преобразует его в код на языке ассемблера.
- Ассемблер формирует объектный код.
- Компоновщик (линкер) производит сборку всех частей в конечные файлы.

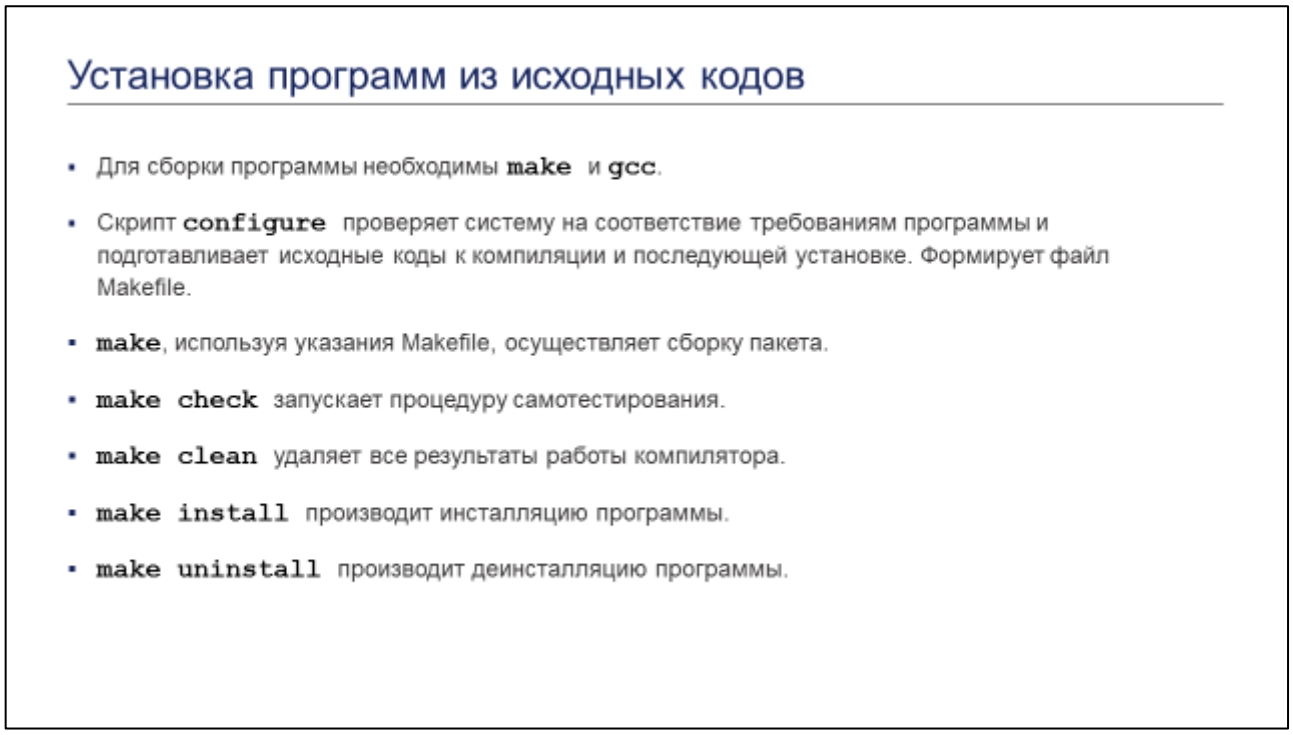
Для создания программы, программист может использовать стандартные библиотеки, предоставляющие необходимый уровень абстракции. Например, основная библиотека **`libc`**, предоставляет функции для работы с памятью, устройствами ввода/вывода. Библиотеки создаются программистами и могут быть статическими, разделяемыми и динамическими:

- Статическая библиотека – код статической библиотеки, при компиляции, встраивается в код программы, тем самым, программа становится самодостаточной, имея в себе все необходимые для работы функции. Имя такой библиотеки имеет расширение "**`.a`**".
- Разделяемая библиотека – код разделяемой библиотеки не встраивается в код программы, загружаясь в память одновременно с запуском программы, в свою очередь, программа получает доступ к необходимым функциям.
- Динамическая библиотека – разновидность разделяемой библиотеки, но она загружается в память только тогда, когда из программы поступит вызов какой-либо функции, находящейся в динамической библиотеке. Имя такой библиотеки имеет расширение "**`.so`**".

Как правило, библиотеки располагаются в каталогах: **/lib**, **/usr/lib**, **/usr/local/lib**. Про все библиотеки, находящиеся в файловой системе, должна знать программа ldconfig, которая производит поиск библиотек во всех каталогах, указанных в файле **/etc/ld.so.conf** и заносит их в свой кэш, впоследствии, компоновщик обращается к **ldconfig** за информацией о необходимых библиотеках.

Обычно программы состоят из множества компонент и при сборке таких продуктов, очень важно отследить, какие файлы уже были откомпилированы, а какие нет, какие файлы необходимо подключать и так далее.

Чтобы управлять сборкой программ была создана утилита **make**, которая определяет необходимость сборки того или иного элемента, основываясь на дате последней модификации. Утилита производит действия, основываясь на файле **Makefile**, в котором есть описание элементов программы, указывающих, от каких элементов они зависят.



Чтобы установить программу, необходимо получить её с сайта разработчика, либо из любого публичного хранилища. Исходные коды поставляются в виде архивных файлов, сжатых программой **gzip** или **bzip2**. Для начала установки программы, распакуйте архив и перейдите в появившийся каталог:

```
# tar xzpf ./mc-4.6.1.tar.gz
# cd mc-4.6.1
```

Теперь необходимо выполнить процедуру конфигурирования в соответствии с вашей системой. Скрипт configure проверит данные системы и соответствие требованиям собираемого пакета. С помощью configure, пользователь может выполнить настройку программы, включить или отключить функции, указать пути для установки, поиска библиотек и т.д. Для просмотра всех возможных параметров, выполните скрипт configure:

```
# configure –help
```

Одним из наиболее часто используемых параметров, является параметр "**--prefix**", который указывает каталог, в который необходимо произвести установку, по умолчанию, как правило, установка произведётся в **/usr/local**. Давайте соберём программу mc, запустим скрипт configure и дождёмся завершения выполнения, если ошибок не будет, то можно приступать к сборке:

```
# configure
# make
# make install
```

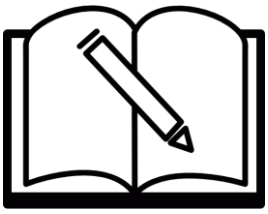
Всё, программа установлена в каталог **/usr/local** и вы можете её запустить, дав команду:

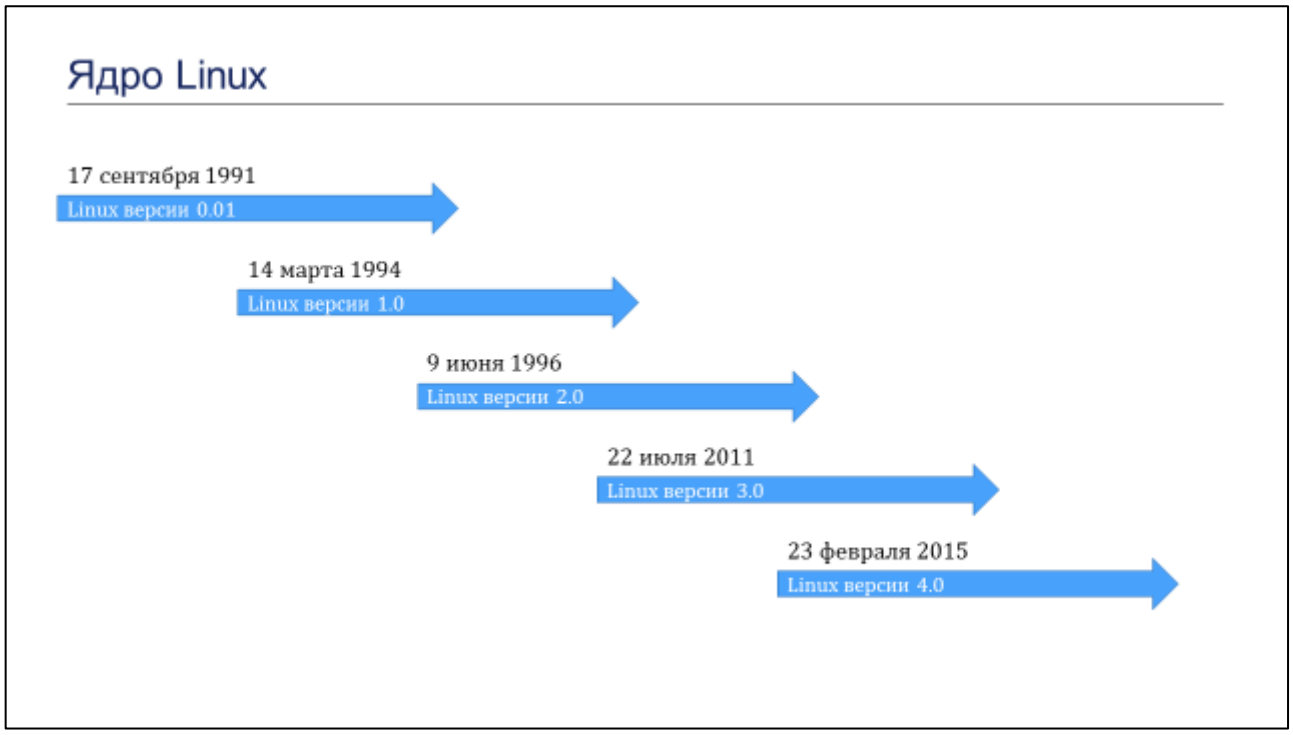
```
# mc
```

Модуль 8. Ядро операционной системы РЕД ОС

В этом модуле рассматриваются:

- Установка и обновление ядра
- Управление параметрами ядра
- Управление модулями ядра





Ядро Linux было создано финским студентом Линусом Торвальдсом. В 1991 году в новостной группе comp.os.minix появилось сообщение следующего содержания:

> *Привет всем, кто использует minix — я делаю (бесплатную) операционную систем (всего лишь хобби, не будет большой и профессиональной как GNU) для клонов 386(486) AT...*

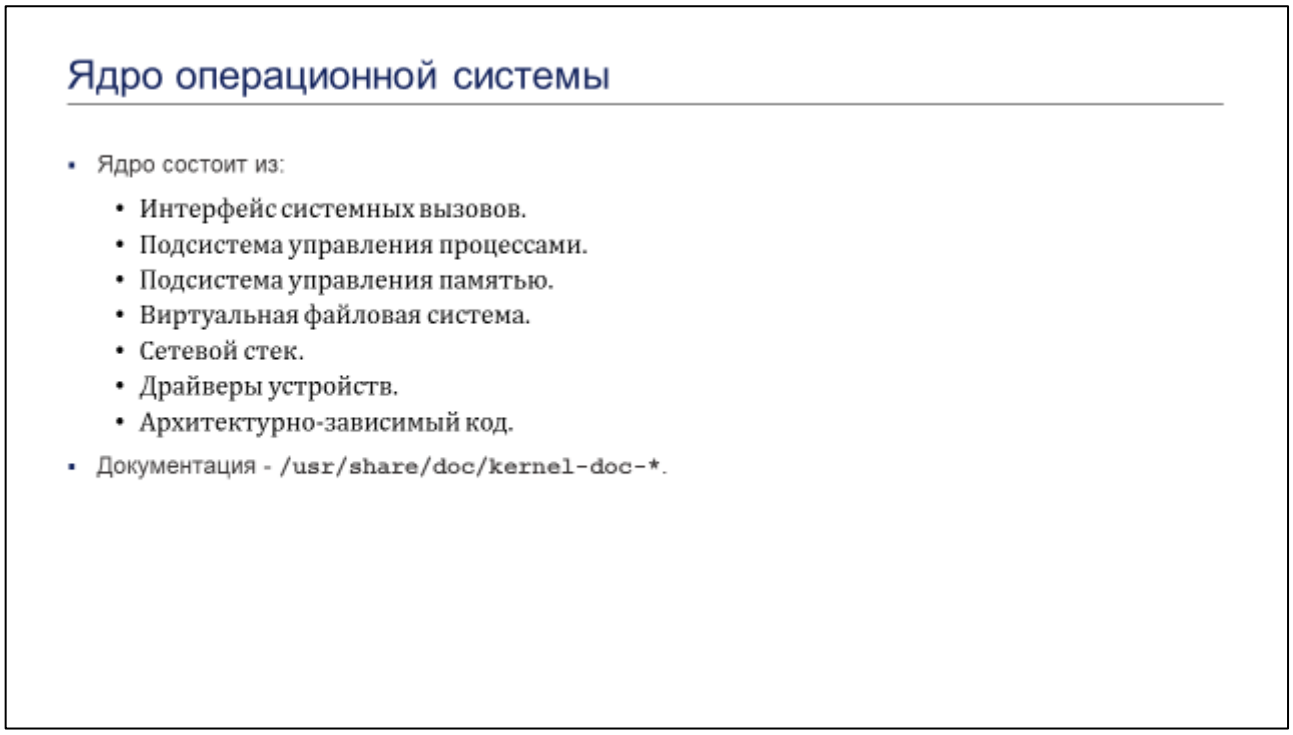
Linux — название ядра, а не операционной системы, но поскольку ядро является основной составляющей операционной системы, то и операционную систему называют Linux. Большинство систем Linux на самом деле необходимо называть GNU/Linux, так как они состоят из ядра Linux и множества программ и библиотек, разработанных в рамках проекта GNU.

В настоящее время, Торвальдс курирует разработку ядра, внося изменения и объединяя изменения, вносимые множеством других программистов со всего мира. Как правило, основные поставщики Linux-дистрибутивов создают свои ответвления от ядра, дополняя их драйверами, которых ещё нет в официальной версии.

Ядро имеет свою нумерацию, состоящую из 3-х чисел — **`A.B[.C]`**:

- Число **`A`** обозначает версию ядра, изменяется только при существенных изменениях в код и концепцию ядра, за всю историю ядра но изменялось трижды.
- Число **`B`** обозначает старшую версию ревизии ядра.
- Число **`C`** обозначает младшую версию ревизии ядра, изменяется при добавлении новых драйверов и улучшений. Если число C отсутствует, это означает, что данная версия ядра является разрабатываемой. Как только разработчики стабилизируют эту версию, она получит номер **`C=0`**, а разрабатываемая версия станет **`A.B+1`**.

Также, часть версий ядра помечены как Long Term Support и поддерживаются дольше, чем обычные версии.



Ядро — очень сложная система, для простоты понимания, разделим его на логические составляющие:

Интерфейс системных вызовов (SCI) — это уровень, предоставляющий средства для вызова функций ядра из пространства пользователей.

Подсистема управления процессами — основой управления процессами является их выполнение. В пространстве ядра процесс называется потоком (thread). Одной из основных задач подсистемы управления процессами является организация совместного использования процессора множеством потоков.

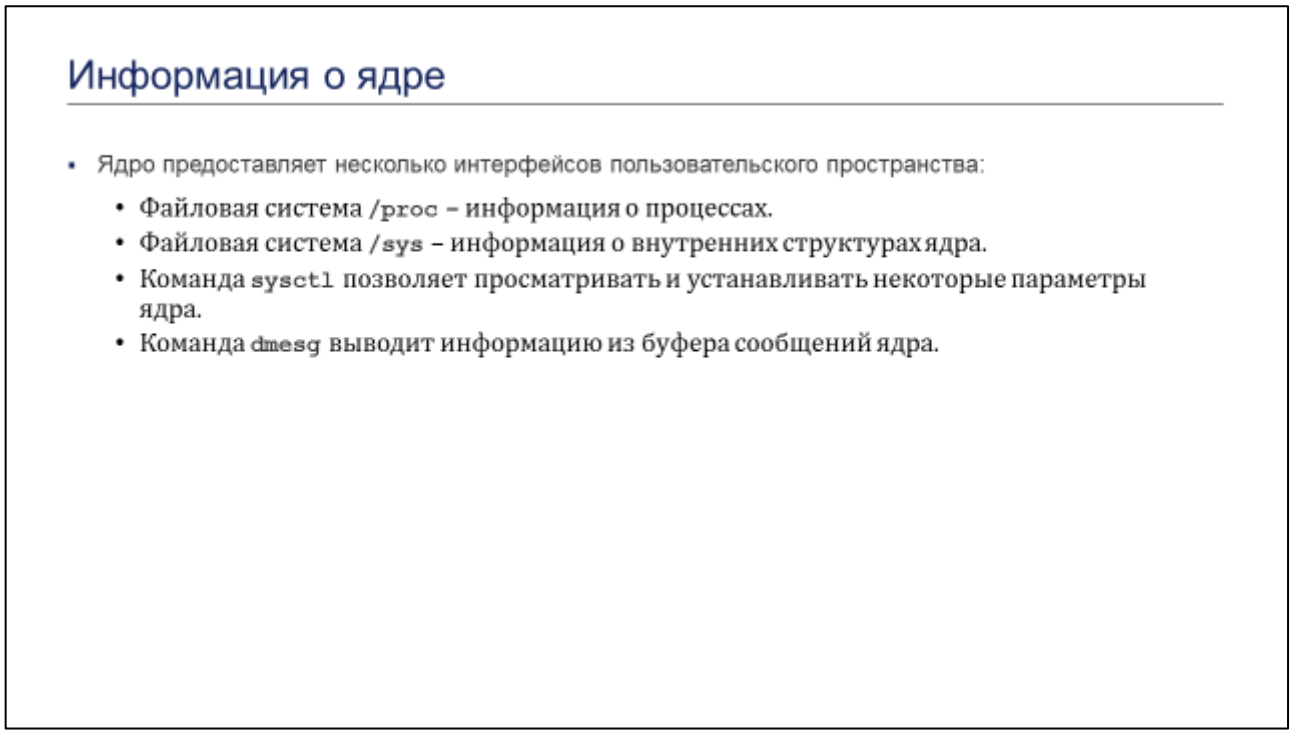
Подсистема управления памятью — ядро, помимо организации работы потоков, не должно забывать и про управление памятью, которая, для повышения эффективности, организуется в виде страниц (для большинства архитектур размер страницы составляет 4 кб). Linux управляет имеющейся памятью и имеет механизмы для установления соответствия между физической и виртуальной памятью. В случае, когда в системе слишком много пользователей памяти, последняя может закончиться, в таком случае, ядро должно вычислить те страницы памяти, которые можно перенести на диск, замедлив работу с ними, но освободив часть оперативной памяти. Такой процесс называется подкачкой.

Виртуальная файловая система — данная подсистема предоставляет общую абстракцию интерфейса к файловым системам. VFS предоставляет уровень коммутации между SCI и файловыми системами, поддерживаемыми ядром. На верхнем уровне VFS располагается API, предоставляющий общую абстракцию к таким функциям файловой системы, как открытие, закрытие, чтение и запись файлов. На нижнем уровне VFS находится сама файловая система.

Сетевой стек — представляет многоуровневую архитектуру, повторяющую структуру самих протоколов. Уровень сокетов представляет собой API к сетевой подсистеме. Он предоставляет пользовательский интерфейс к различным сетевым протоколам.

Драйверы устройств — большей частью исходного кода ядра являются коды драйверов устройств, обеспечивающих возможность работы с различными устройствами.

Архитектурно-зависимый код — основная часть ядра независима от архитектуры, на которой работает операционная система. Тем не менее, некоторые части требуют привязки к архитектуре системы для того, чтобы повысить производительность.



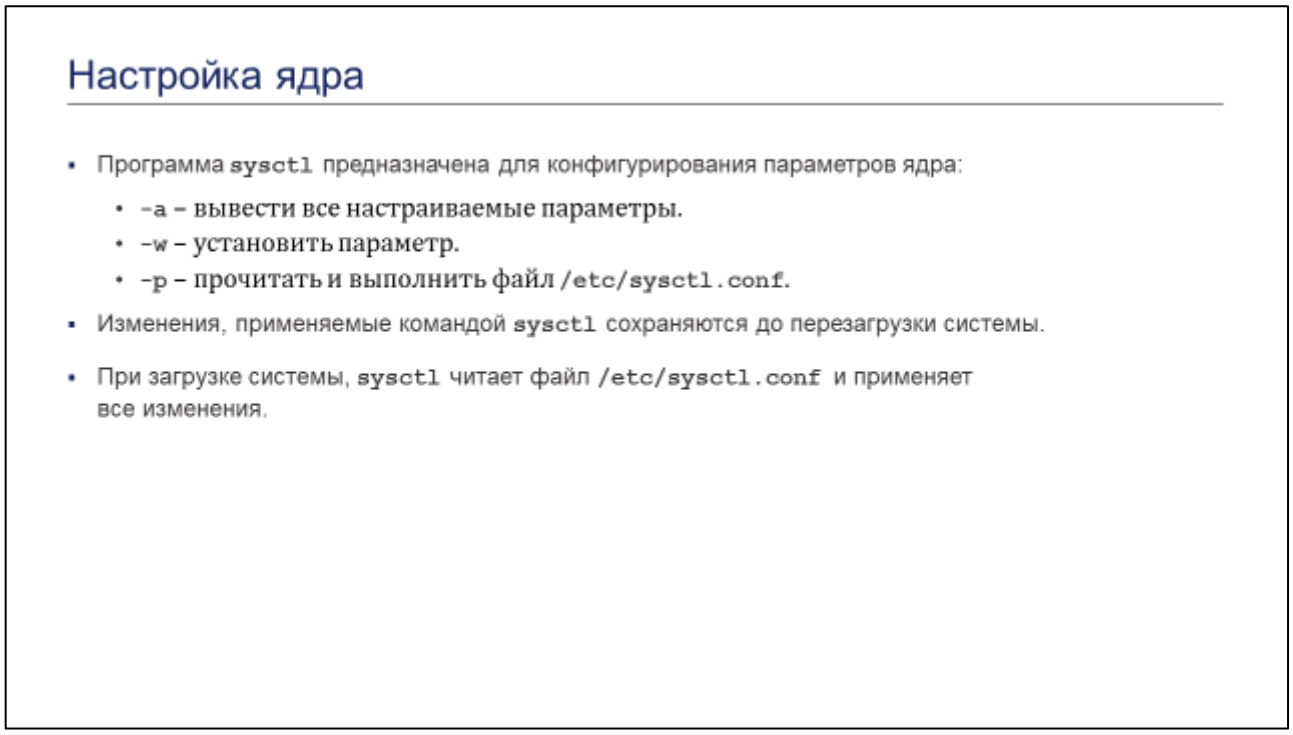
Виртуальная файловая система **/proc** предоставляет доступ к детальной информации о жизнедеятельности ядра, запущенных процессах. Каждый файл содержит информацию о каком-либо элементе.

В **/proc** есть несколько очень интересных файлов:

- **/proc/cmdline** – команды, которые были переданы загрузчику в командной строке.
- **/proc/cpuinfo** – информация о процессоре.
- **/proc/meminfo** – информация об использовании памяти.
- **/proc/modules** – список загруженных модулей.
- **/proc/partitions** – информация о разделах.
- **/proc/net/arp** – ARP таблица.
- **/proc/net/route** – таблица маршрутизации.

Файловая система **/sys** появилась относительно недавно и призвана вытеснить устаревшую файловую систему **/proc**:

- **/sys/block/*** – информация о блочных устройствах;
- **/sys/bus/*** – информация об устройствах;
- **/sys/modules/*** – информация о модулях.

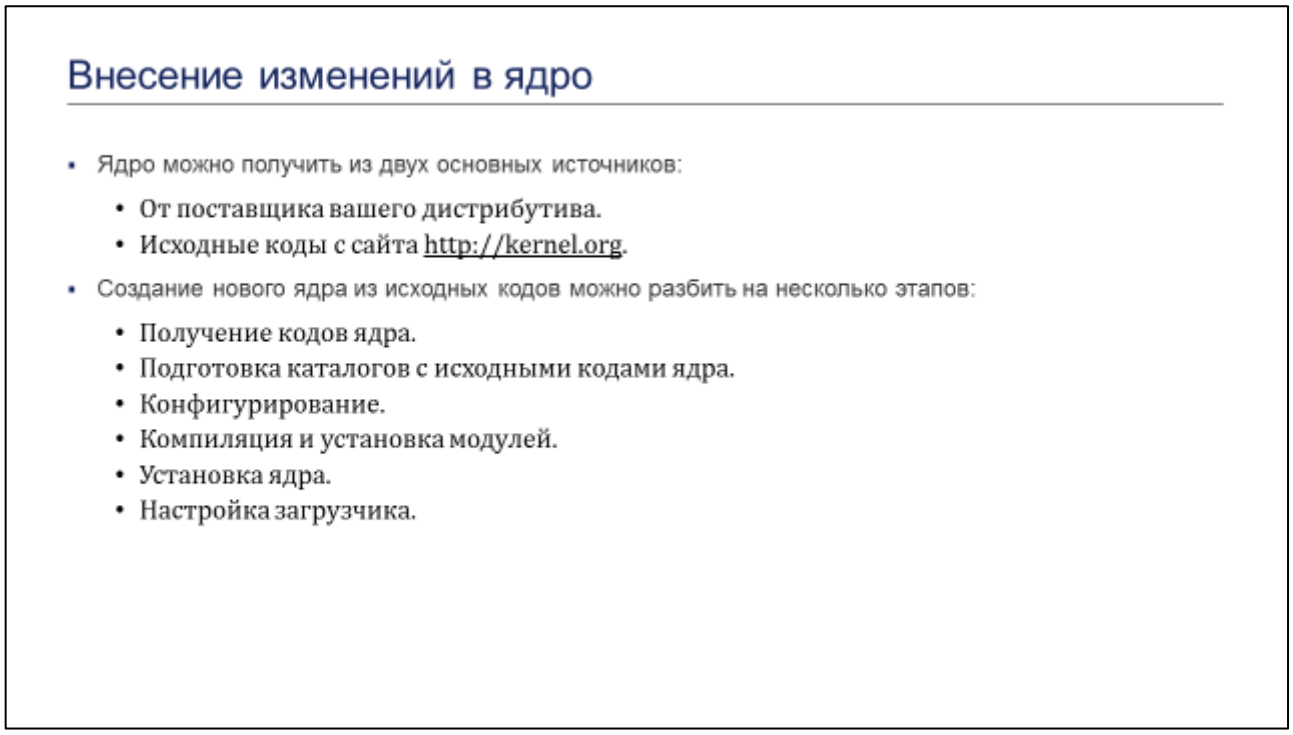


Ядро, поставляемое с системой настроено таким образом, чтобы удовлетворять среднестатистические потребности, но в процессе работы вам может понадобиться более тонкая настройка некоторых параметров ядра. Документация, поставляемая с ядром, содержит описание параметров. Команда **sysctl** предоставляет администратору системы возможность настройки тех параметров, которые ему кажутся не оптимальными. Для получения списка всех параметров, воспользуйтесь командой:

```
# sysctl -a
```

Чтобы установить параметр, используйте параметр **-w**. Например, чтобы включить перенаправление сетевых пакетов, можно воспользоваться двумя разными командами, приводящими в одному и тому же результату:

```
# sysctl -w net.ipv4.ip_forward=1
# echo 1 > /proc/sys/net/ipv4/ip_forward
```



Как правило ядро, поставляемое с дистрибутивом операционной системы, включает в себя полный набор всех драйверов, которые могут понадобиться для установки на различные системы. Естественно, что для каждой системы лучше включить в ядро только то, что действительно необходимо, тем самым избежав лишних потерь производительности.

Чтобы собрать ядро необходимо его получить. Все версии ядер находятся на сервере kernel.org, откуда вы можете свободно загрузить исходные коды ядра.

Чтобы получить исходные коды, воспользуйтесь командной **wget**:

```
# cd /usr/src
# wget http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.22.9.tar.bz2
```

Теперь необходимо перейти в распакованный каталог с исходными кодами ядра Linux и запустить командой make программу конфигурирования. Существует несколько способов настройки ядра:

- **menuconfig** — цветное текстовое меню с диалоговыми окнами, является основным способов настройки ядра.
- **xconfig** — то же самое, что и menuconfig, но в GUI (X-сервер). Основано на библиотеке Qt.
- **gconfig** — то же самое, что и xconfig, но основано на библиотеке Gtk.
- **oldconfig** — конфигурирование основывается на настройках в файле **./.config**. Пользователь должен настроить только новые функции.
- **allmodconfig** — конфигурирование будет произведено таким образом, что везде, где это возможно, будет использована модульная структура.

Основным конфигурационным файлом является файл **.config**, который появляется в результате произведённых настроек. По умолчанию, во многих дистрибутивах, в каталоге **/boot** помимо файла ядра находится его конфигурация:

```
# ls /boot
…
config-2.6.21-1.3194.fc7            vmlinuz-2.6.21-1.3194.fc7
…
```

В случае, когда вы хотите произвести обновление ядра с использованием текущей конфигурации, скопируйте конфигурационный файл в дерево исходных кодов ядра и выполните команду:

```
# cp config-2.6.21-1.3194.fc7 /usr/src/linux-2.6.22.9/.config
# cd /usr/src/linux-2.6.22.9/
# make oldconfig
```

После выполнения этой команды начнётся пройцесс конфигурирования ядра, при котором на все вопросы конфигуратора ответы будут получены из файла **./.config**, на все вопросы, касаемо новых функций ядра, будет предложено ответить администратору.

Порой необходимо не только обновить ядро, но и внести какие-то изменения, будь то отключение ненужных модулей или изменение типа ядра. Чтобы настроить ядро так, как вам требуется, дайте следующую команду:

```
# make menuconfig
```

Всё, что вам необходимо сделать, отметить необходимые пункты, в соответствии с подсказкой, данной в верхней части меню.

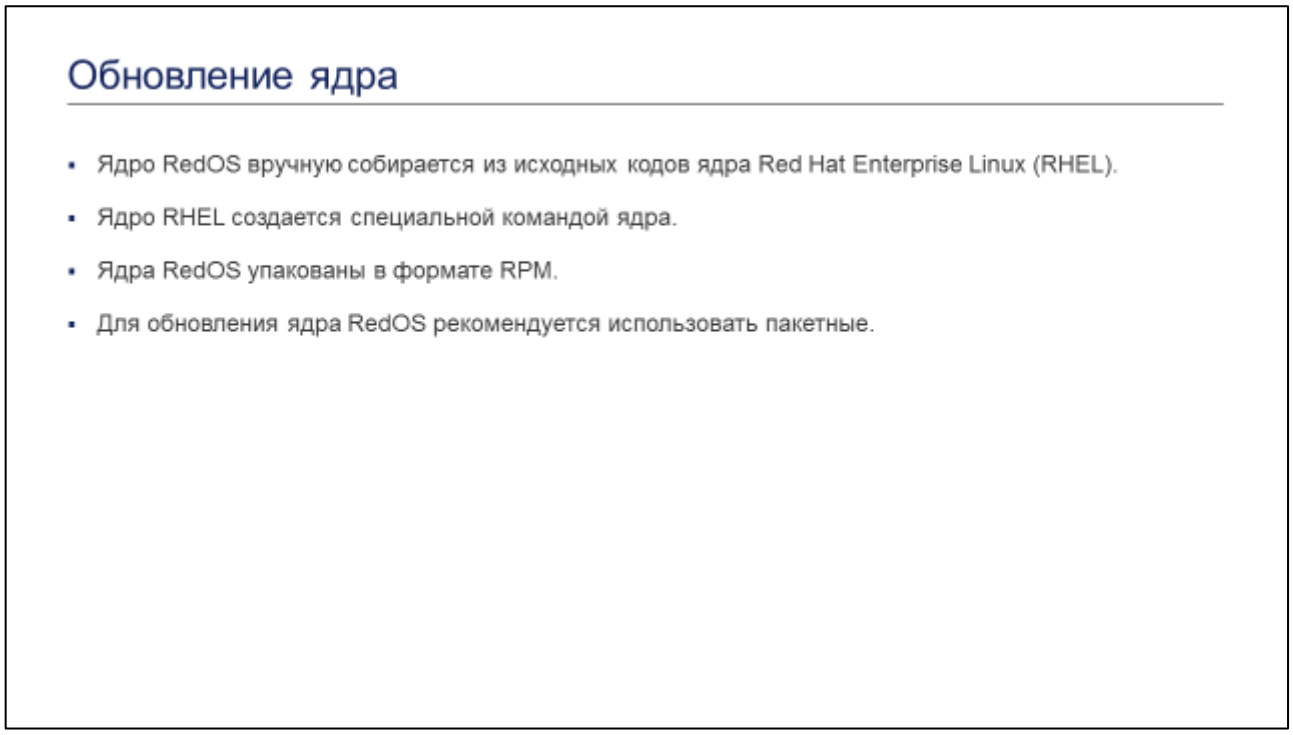
После того, как ядро было настроено необходимо произвести компиляцию и установку. Для компиляции необходимо выполнить, уже известную, команду make, после чего начнётся процесс сборки, который может затянуться на некоторое время. Если компиляция была завершена без ошибок, значит, что ядро было собрано (**/usr/src/linux-2.6.32.59/arch/i386/boot/bzImage**), его уже можно использовать, но перед этим требуется установить все модули в каталог **/usr/lib**, выполните команду:

```
# make
# make modules_install
```

Теперь в каталоге /usr/lib появилось дерево модулей, необходимых для работы нового ядра. Остаётся только скопировать файл ядра в каталог /boot и внести соответствующие изменения в конфигурационный файл загрузчика grub. Также, возможна автоматическая установка ядра:

```
# make install
```

Не забывайте, что новое ядро может не загрузиться по каким-либо причинам, поэтому всегда оставляйте возможность загрузки с предыдущей версией ядра.



Ядро RedOS вручную собирается из исходных кодов ядра Red Hat Enterprise Linux (RHEL). В свою очередь ядро RHEL создается специальной командой ядра, чтобы убедиться в целостности и совместимости с поддерживаемым оборудованием. Перед каждым релизом ядра, Red Hat проводит комплексное оценочное тестирование.

Ядра RedOS упакованы в формате RPM, таким образом, что они могут быть легко обновлены и проверены при помощи пакетного менеджера Yum.

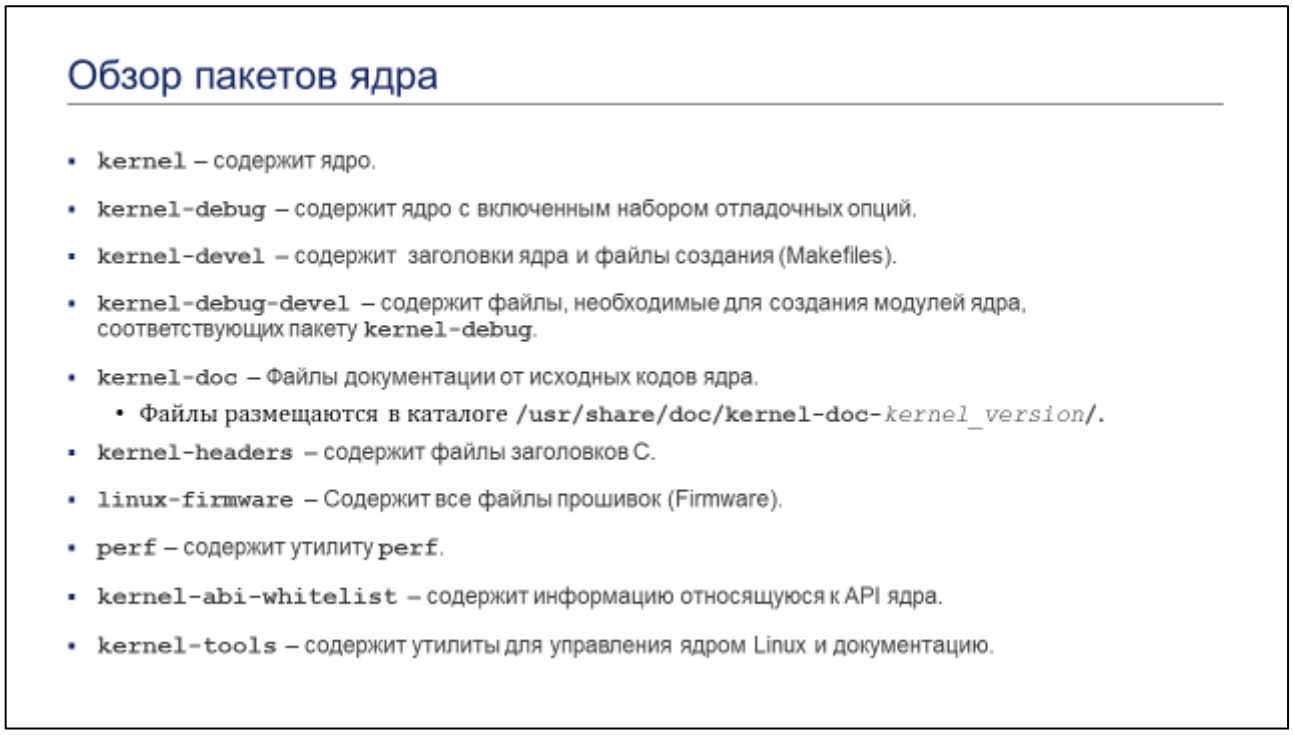
Далее мы детально разберем управляемый процесс обновления пакета ядра при помощи команды **`rpm`** вместо **`yum`**.

ПРЕДУПРЕЖДЕНИЕ

По возможности используйте пакетный менеджер Yum для установки новых версий ядра, так как он устанавливает новое ядро, вместо замены текущего, что позволяет защититься от невозможности загрузиться.

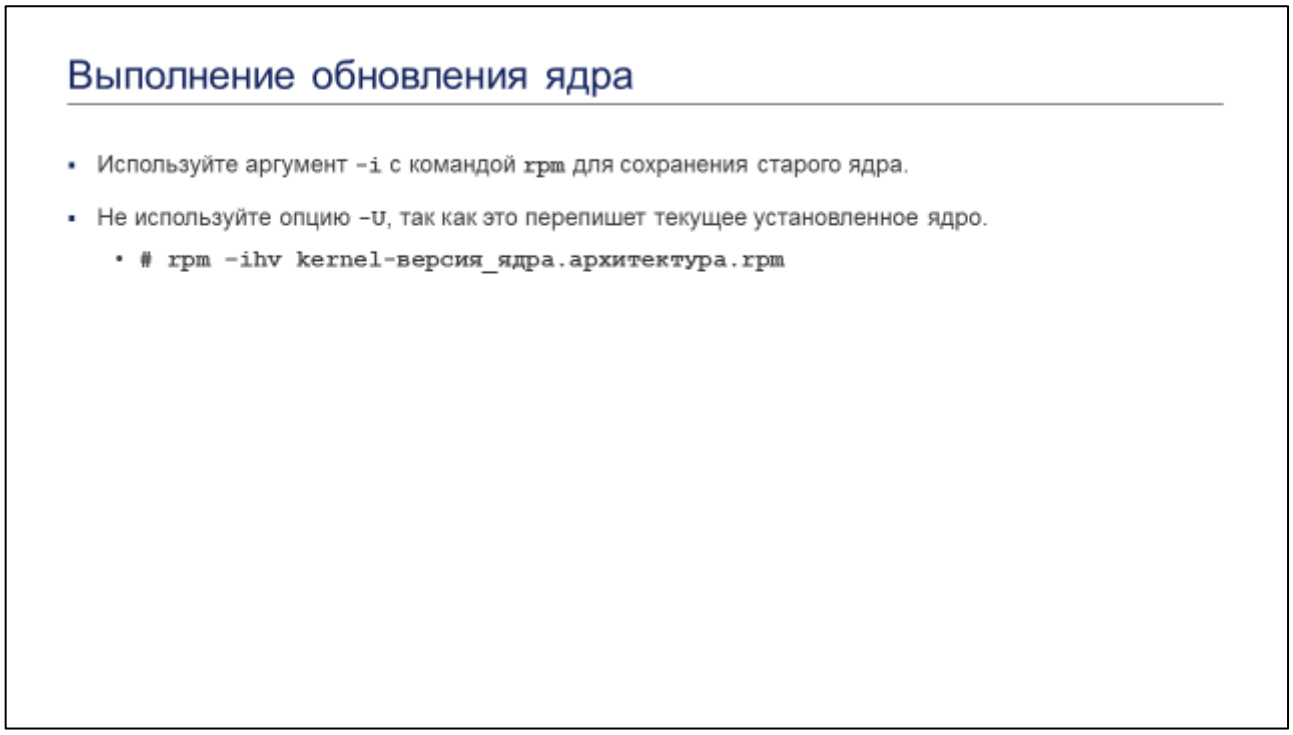
ПРЕДУПРЕЖДЕНИЕ

Сборка ядра из исходных кодов не рекомендуется для дистрибутива RedOS.



К ядро операционной системы относятся следующие пакеты:

- **kernel** – содержит ядро, для одноядерных, многоядерных и многопроцессорных систем.
- **kernel-debug** – содержит ядро с включенным набором отладочных опций для диагностики ядра и с меньшей производительностью.
- **kernel-devel** – содержит заголовки ядра и файлы создания (Makefiles) необходимые для создания модулей ядра.
- **kernel-debug-devel** – содержит файлы, необходимые для создания модулей ядра, соответствующих пакету kernel-debug.
- **kernel-doc** – Файлы документации от исходных кодов ядра. Различные секции ядра Linux и поставляемых драйверов устройств описаны в этих файлах документации. Установка этих файлов предоставляет описание к опциям, которые могут быть переданы модулям ядра Linux во время загрузки.
- По умолчанию эти файлы размещаются в каталоге **/usr/share/doc/kernel-doc-***kernel_version***/**.
- **kernel-headers** – содержит файлы заголовков C, которые определяют интерфейс между ядром Linux и пользовательскими библиотеками, и программами. Файлы заголовков определяют структуру и константы необходимые для создания большинства стандартных программ.
- **linux-firmware** – Содержит все файлы прошивок (Firmware), которые необходимы различным устройствам для работы.
- **perf** – этот пакет, содержит утилиту **perf**, которая обеспечивает мониторинг ядра Linux.
- **kernel-abi-whitelist** – содержит информацию, относящуюся к API ядра RedOS (RHEL), в том числе список символов ядра, которые необходимы внешним модулям ядра Linux и плагину yum для обеспечения применения.
- **kernel-tools** – содержит утилиты для управления ядром Linux и документацию.



После загрузки всех необходимых пакетов, пришло время для обновления существующего ядра.

ПРИМЕЧАНИЕ

Настоятельно рекомендуется сохранить старое ядро на случай возникновения проблем с новым ядром.

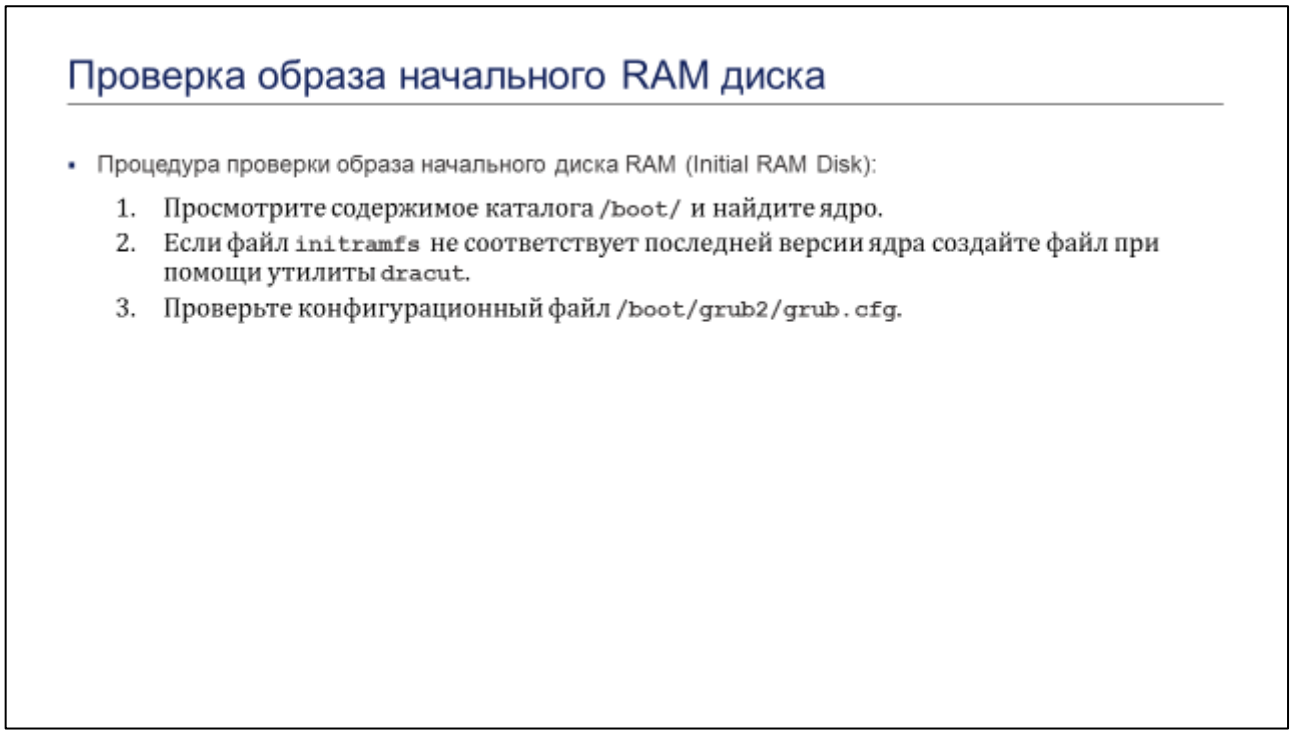
В командной строке перейдите в каталог с загруженным RPM пакетом. Используйте аргумент -i с командой rpm для сохранения старого ядра. Не используйте опцию -U, так как это перепишет текущее установленное ядро, что может в дальнейшем вызвать проблемы с загрузкой.

```
# rpm -ihv kernel-версия_ядра.архитектура.rpm
```

Пример установки нового ядра:

```
# rpm -ihv kernel-3.10.0-327.36.3.el7.x86_64.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:kernel-3.10.0-327.36.3.el7       ################################# [100%]
```

Следующим шагом необходимо проверить, что начальный диск RAM (Initial RAM Disk) был создан.



Основная задача начального RAM диска заключается в загрузку модулей блочных устройств, таких как IDE, SCSI или RAID, так как корневая файловая система обычно располагается на них, они должны быть доступны и смонтированы. В системах RedOS 7 (RHEL 7), когда новое ядро устанавливается при помощи пакетных менеджеров Yum или RPM, утилита Dracut всегда вызывает установочный скрипт для создания **initramfs** (образ начального диска RAM).

На всех архитектурах кроме IBM eServer System i можно создать **initramfs** при помощи утилиты **dracut**. Однако, обычно нет необходимости создавать **initramfs** вручную, так как этот шаг выполняется автоматически если ядро и связанные пакеты установлены или обновляются из пакетов RPM.

Вы можете проверить, что **initramfs** соответствует текущей версии ядра и он корректно указан в конфигурационном файле **grub.cfg**, при помощи следующей процедуры:

Процедура проверки образа начального диска RAM (Initial RAM Disk).

1. От имени пользователя root просмотрите содержимое каталога **/boot/** и найдите ядро (**vmlinuz-***версия_ядра*) и **initramfs-***версия_ядра* с последними версиями.

```
# ls /boot/
config-3.10.0-327.22.2.el7.x86_64
config-3.10.0-327.36.3.el7.x86_64
grub
grub2
initramfs-0-rescue-bef274407d4d409ab35ad045ffa59ab3.img
initramfs-3.10.0-327.22.2.el7.x86_64.img
initramfs-3.10.0-327.36.3.el7.x86_64.img
initrd-plymouth.img
symvers-3.10.0-327.22.2.el7.x86_64.gz
symvers-3.10.0-327.36.3.el7.x86_64.gz
System.map-3.10.0-327.22.2.el7.x86_64
System.map-3.10.0-327.36.3.el7.x86_64
vmlinuz-0-rescue-bef274407d4d409ab35ad045ffa59ab3
vmlinuz-3.10.0-327.22.2.el7.x86_64
```

```
vmlinuz-3.10.0-327.36.3.el7.x86_64
```

Предыдущий пример показывает:

- Одно или несколько ядер установлены (если точнее, то следующие файлы представлены в каталоге **/boot**).
- Последняя версия ядра это **vmlinuz-3.10.0-327.36.3.el7.x86_64**.
- Файл **initramfs**, соответствующий версии ядра, **initramfs-3.10.0-327.36.3.el7.x86_64.img** также существует.

ПРИМЕЧАНИЕ

В каталоге **/boot/** вы можете найти несколько файлов **initrd-***версия***kdump.img**. Это специальные файлы, созданные механизмом Kdump в целях отладки ядра, не используются для загрузки системы и могут быть безопасно проигнорированы.

2. Если файл **initramfs-***версия_ядра* не соответствует последней версии ядра в каталоге **/boot/**, или в иных подобных ситуациях, вам может потребоваться создать файл **initramfs** при помощи утилиты **dracut**. В таком случае просто вызовите **dracut** с правами пользователя root, без указания параметров для создания файла **initramfs** в каталоге **/boot/** для последней версии ядра, представленной в каталоге:

```
# dracut
```

Необходимо использовать опцию **-f**, **--force**, если вы хотите переписать существующие версии initramfs (например, когда initramfs поврежден). Иначе **dracut** проигнорирует существующие файлы **initramfs**:

```
# dracut
Will not override existing initramfs (/boot/initramfs-3.10.0-
327.22.2.el7.x86_64.img) without --force
```

Вы можете создать **initramfs** в текущем каталоге при помощи **dracut**.

```
# dracut "initramfs-$(uname -r).img" $(uname -r)
```

Если необходимо указать определенные модули ядра, чтобы они были предварительно загружены, нужно добавить имена этих модулей (без указания каких-либо суффиксов, таких как **.ko**) в директиве **add_dracutmodules+="***модуль* **[***дополнительные_модули***]"** конфигурационного файла **/etc/dracut.conf**. Вы можете просмотреть содержимое образа **initramfs** созданного утилитой dracut, при помощи команды **lsinitrd** *файл_initramfs*:

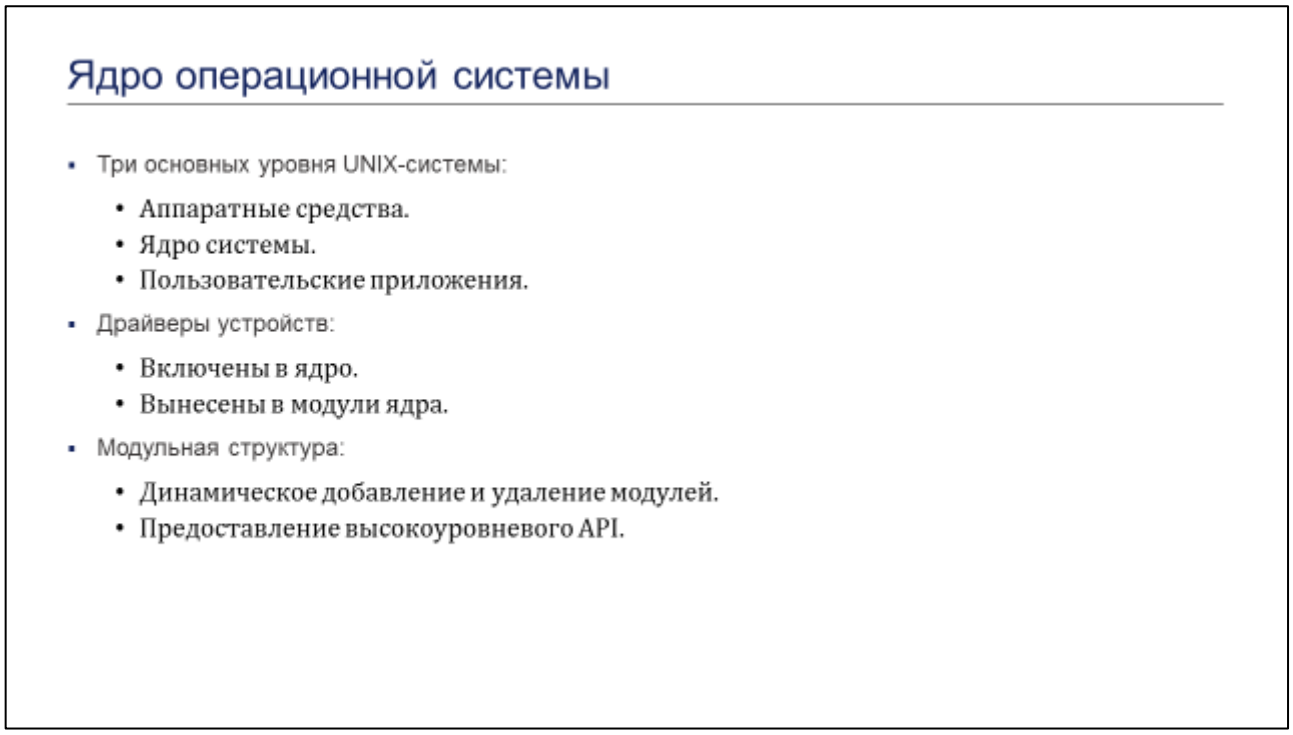
```
# lsinitrd /boot/initramfs-3.10.0-327.36.3.el7.x86_64.img
Image: /boot/initramfs-3.10.0-327.36.3.el7.x86_64.img: 29M
========================================================================
Early CPIO image
========================================================================
drwxr-xr-x   3 root     root            0 Dec  4 08:29 .
-rw-r--r--   1 root     root            2 Dec  4 08:29 early_cpio
drwxr-xr-x   3 root     root            0 Dec  4 08:29 kernel
drwxr-xr-x   3 root     root            0 Dec  4 08:29 kernel/x86
drwxr-xr-x   2 root     root            0 Dec  4 08:29 kernel/x86/microcode
-rw-r--r--   1 root     root        10240 Dec  4 08:29
kernel/x86/microcode/GenuineIntel.bin
========================================================================
```

```
Version: dracut-033-360.el7_2.1
…
```

Для получения дополнительной информации по опциям и использованию воспользуйтесь **man dracut** и man **dracut.conf**.

3. Проверьте конфигурационный файл **/boot/grub2/grub.cfg** чтобы убедиться, что **initrd initramfs-***версия_ядра***.img** существует для версии ядра, которое необходимо загрузить. Например:

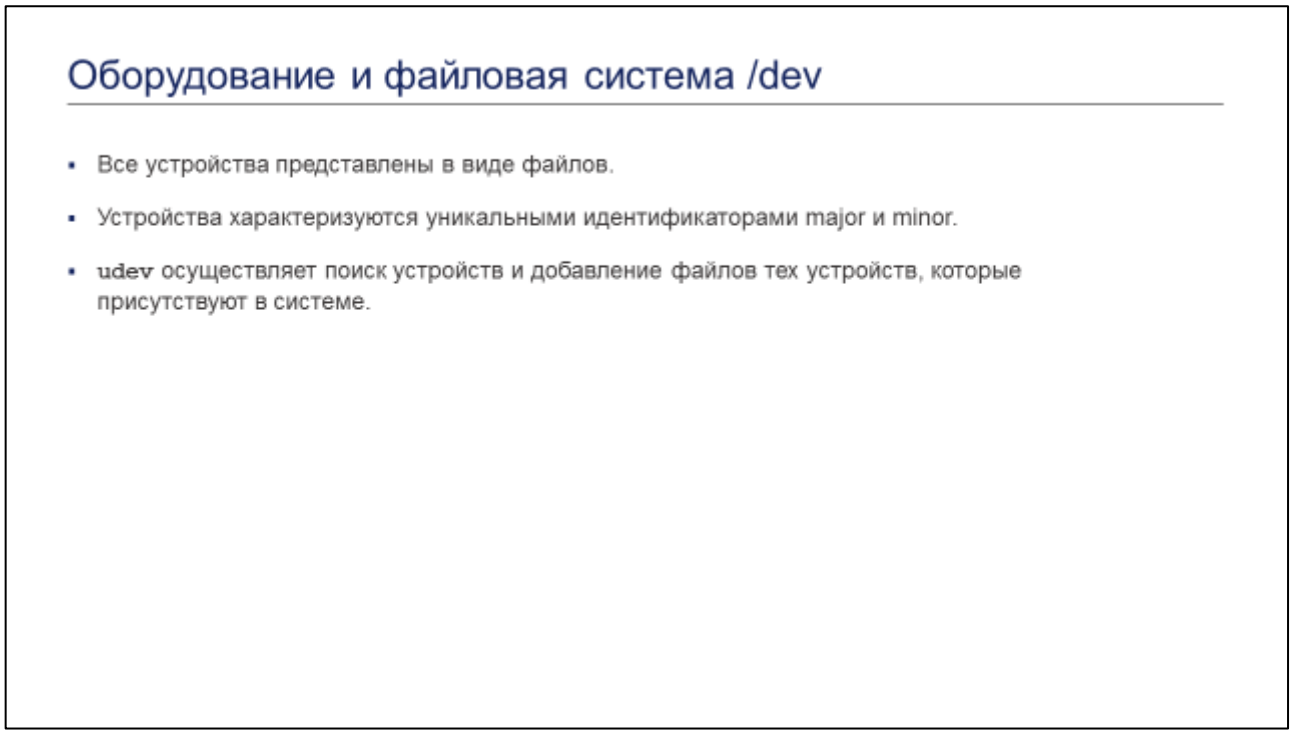
```
# grep initramfs /boot/grub2/grub.cfg
        initrd16 /initramfs-3.10.0-327.36.3.el7.x86_64.img
        initrd16 /initramfs-3.10.0-327.22.2.el7.x86_64.img
        initrd16 /initramfs-0-rescue-bef274407d4d409ab35ad045ffa59ab3.img
```



Большая часть ядра Linux – это драйверы, с помощью которых происходит процесс общения с аппаратными устройствами. Ядро скрывает этот процесс, предоставляя высокий уровень абстракции с помощью API, который предоставляет программистам и пользователям некоторые общие концепции, присущие определённому типу устройств, тем самым освобождая программы от задачи общения с разнообразными устройствами.

Ядро содержит драйверы устройств, которые управляют отдельными элементами аппаратного уровня. Учитывая, что Linux поддерживает огромное множество самого разнообразного оборудования, встаёт вопрос о размере ядра и его быстродействии. Ответом на этот вопрос служит модульная структура ядра. Драйверы могут быть вынесены в отдельные модули, что позволяет оставить в ядре только самый необходимый функционал, а, следовательно, уменьшить размер ядра и увеличить его быстродействие.

Модули могут быть загружены и выгружены динамически, не требуя останова системы.



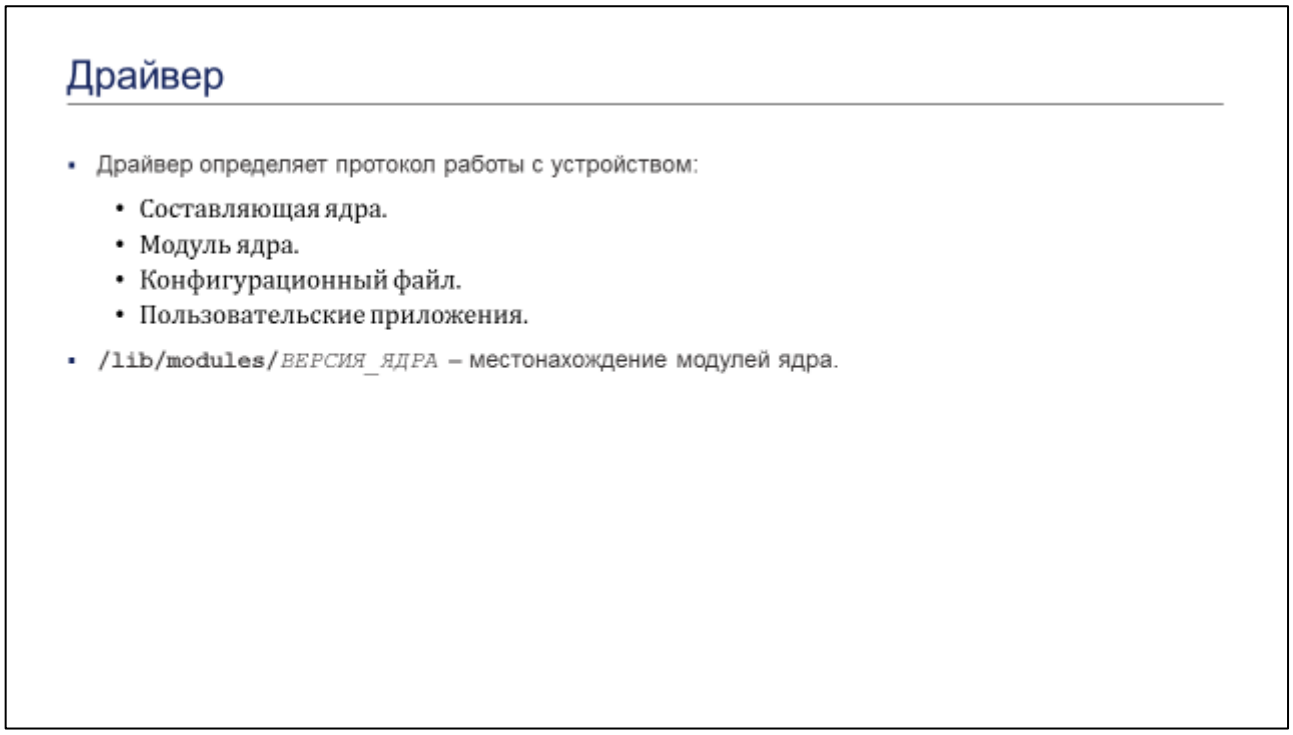
Драйвер устройства – это программа, которая обеспечивает взаимодействие системы с определённым компонентом аппаратной среды. Драйвер выполняет роль переводчика команд конкретного устройства на "язык" API-функций ядра.

Драйверы являются компонентами ядра, а не пользовательскими процессами. Несмотря на это, ядро предоставляет доступ к устройствам программам пользовательского уровня. Для этого существует специальная виртуальная файловая система **/dev**, в которой создаются специальные файлы устройств, ядро преобразует все операции над этими файлами в обращения к драйверу.

Каждый файл устройства в каталоге **/dev** имеет старший и младший номер устройства (major и minor). Старший номер устройства обозначает драйвер, к которому относится этот файл, а младший обозначает конкретный тип устройства. Посмотреть номера устройств можно с помощью команды **ls**, в поле, в котором для обычных файлов указывается размер, будут указаны номера устройства:

```
$ ls -l /dev/sda /dev/sda1
brw-r----- 1 root disk 8, 0 Sep 16 10:45 /dev/sda
brw-r----- 1 root disk 8, 1 Sep 16 10:46 /dev/sda1
```

Существует два типа специальных файлов: блок-ориентированные и байт-ориентированные. Чтение из блок-ориентированного устройства производится по одному блоку (как правило, кратному 512) за раз, тогда как чтение из байт-ориентированного устройства производится побайтно



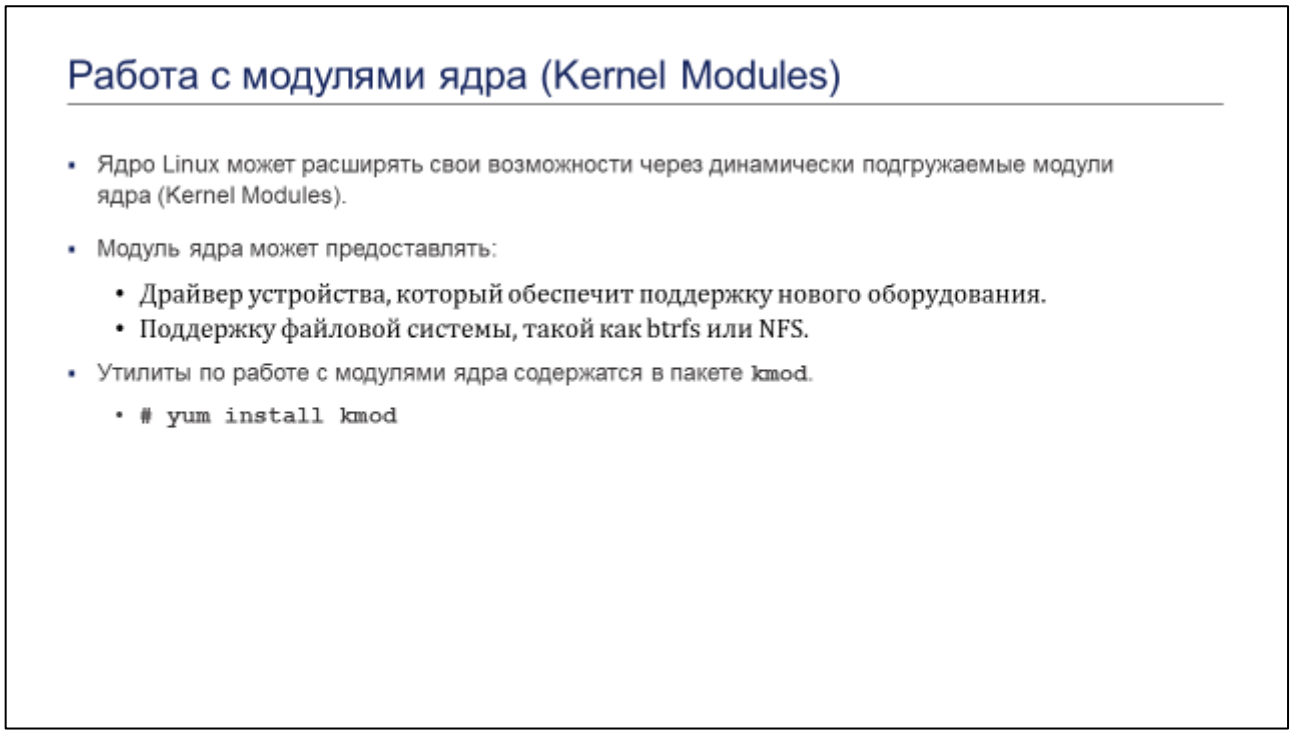
Драйвер – неотъемлемая часть операционной системы, благодаря драйверам ядро может "разговаривать" с устройствами на одном языке.

Ядро Linux включает в себя огромное множество драйверов, позволяющих работать с самым разнообразным оборудованием. Учитывая модульную структуру ядра, совсем не обязательно расходовать ресурсы на всевозможные драйверы, из которых необходимы единицы. Драйверы могут быть вынесены в отдельные модули, загружаемые ядром по мере необходимости.

Не всегда драйверы являются составляющей частью ядра. Протоколом, определяющим работу с устройством, может быть и простой текстовый файл, например, PPD-файлы, определяющие протокол взаимодействия с принтерами.

Все драйверы (модули ядра) находятся в каталоге **/lib/kernel/***ВЕРСИЯ_ЯДРА*:

```
$ ls -l /lib/modules/2.6.32-279.el6.i686/kernel
drwxr-xr-x  3 root root 4096 Aug  9 10:34 arch
drwxr-xr-x  2 root root 4096 Aug  9 10:34 crypto
drwxr-xr-x 40 root root 4096 Aug  9 10:34 drivers
drwxr-xr-x 45 root root 4096 Aug  9 10:34 fs
drwxr-xr-x  4 root root 4096 Aug  9 10:34 lib
drwxr-xr-x 25 root root 4096 Aug  9 10:34 net
drwxr-xr-x  9 root root 4096 Aug  9 10:34 sound
```



Ядро Linux является модульным, это значит, оно может расширять свои возможности через динамически подгружаемые модули ядра (Kernel Modules).

Модуль ядра может предоставлять:

- Драйвер устройства, который обеспечит поддержку нового оборудования.
- Поддержку файловой системы, такой как btrfs или NFS.

Как и само ядро, модули могут принимать параметры, которые настраивают их поведение. В большинстве случаев параметры по умолчанию работают достаточно хорошо, чтобы не пришлось их менять. Инструменты пользовательского пространства могут выводить список модулей, загруженных в запущенное ядро; запрашивать все доступные модули, доступные параметры модулей и информацию о модуле; динамически подгружать и выгружать (удалять) модули в или из запущенного ядра. Многие утилиты, которые предоставляются пакетом **kmod**, обладают способностью обрабатывать зависимости модулей при выполнении операций, что не требует ручного отслеживания зависимостей.

На современных системах модули ядра автоматически загружаются при помощи различных механизмов при появлении потребности в их вызове. Однако, возможны ситуации, при которых необходимо загружать и выгружать модули вручную, например, когда один модуль предпочтительнее другого, хотя оба предоставляют одинаковую функциональность, или, когда модуль функционирует некорректно.

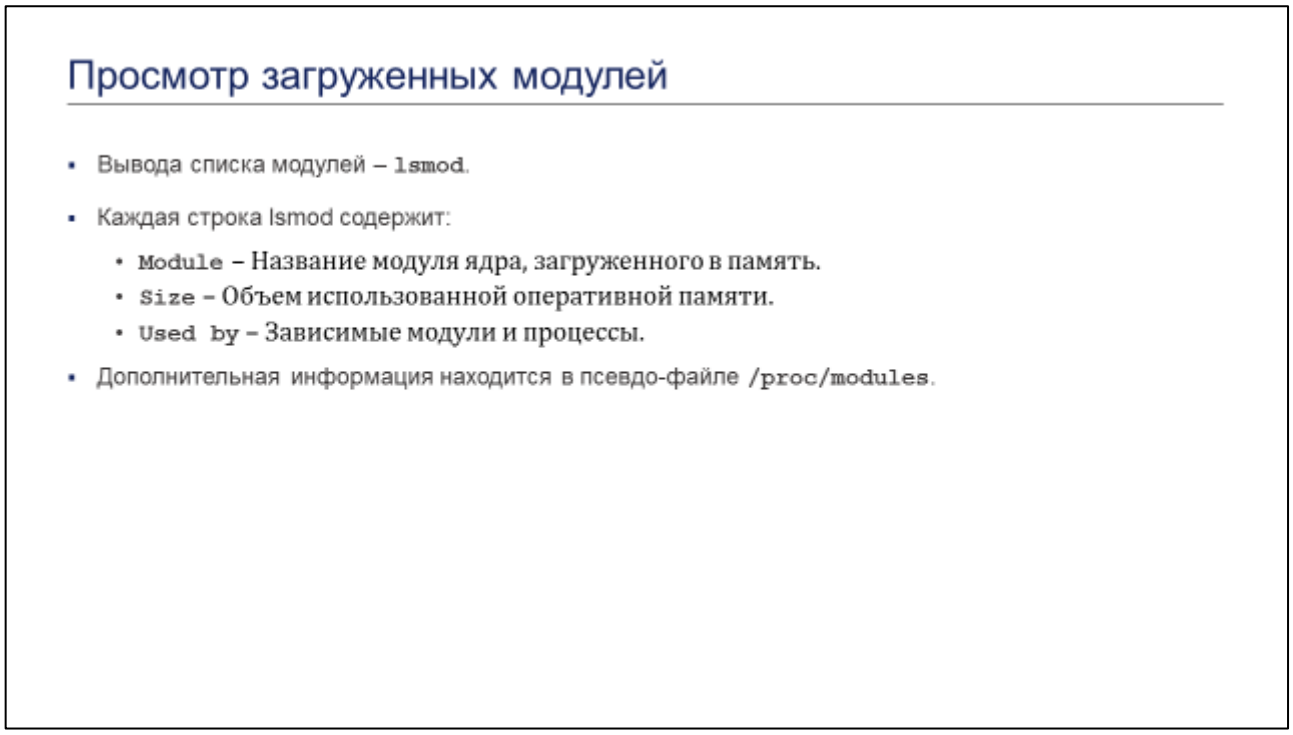
Далее будет рассмотрено следующее:

- Использование утилит пользовательского пространства **kmod** для отображения, запросов, загрузки и выгрузки модулей ядра и их зависимостей.
- Динамическая установка параметров при помощи командной строки и на постоянной основе, для настройки поведения модулей.
- Загрузка модулей ядра в процессе загрузки системы.

ПРИМЕЧАНИЕ

Для того чтобы использовать утилиты по работе с модулями ядра, описанные далее, в первую очередь необходимо убедиться, что пакет **kmod** установлен на системе:

```
# yum install kmod
```



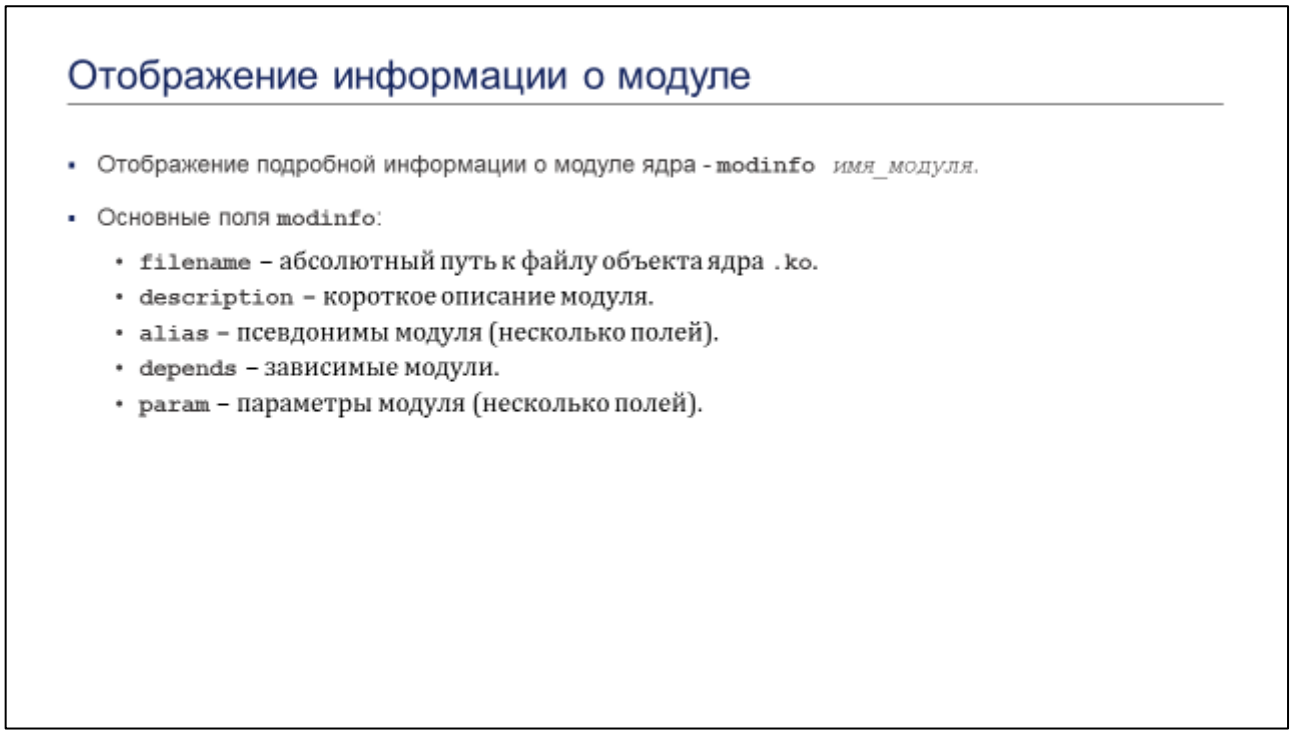
Для вывода списка модулей, которые загружены в ядро, воспользуйтесь командой **lsmod**:

```
# lsmod
Module                  Size  Used by
ip6t_rpfilter          12546  1
ip6t_REJECT            12939  2
ipt_REJECT             12541  2
xt_conntrack           12760  23
ebtable_nat            12807  0
ebtable_broute         12731  0
bridge                115385  1 ebtable_broute
stp                    12976  1 bridge
llc                    14552  2 stp,bridge
ebtable_filter         12827  0
ebtables               30913  3 ebtable_broute,ebtable_nat,ebtable_filter
ip6table_nat           12864  1
nf_conntrack_ipv6      18738  13
nf_defrag_ipv6         34651  1 nf_conntrack_ipv6
nf_nat_ipv6            14131  1 ip6table_nat
ip6table_mangle        12700  1
ip6table_security      12710  1
ip6table_raw           12683  1
ip6table_filter        12815  1
…
```

Каждая строка lsmod содержит:

- **Module** – Название модуля ядра, загруженного в память.
- **Size** – Объем использованной оперативной памяти.
- **Used by** – Количество процессов, которые используют модуль и прочие модули, зависящие от этого, далее следует список имен этих модулей. При помощи этого списка, вы можете в первую очередь выгрузить все модули, зависящие от модуля, который вы хотите выгрузить.

Вывод **lsmod** менее подробный и более простой для восприятия, чем содержимое псевдо-файла **/proc/modules**.



Вы можете отобразить подробную информацию о модуле ядра при помощи команды **modinfo** *имя_модуля*.

ПРИМЕЧАНИЕ

При вводе имени модуля в качестве аргумента утилиты **kmod**, не нужно добавлять расширение **.ko** в конец имени. Имена модулей ядра не имеют расширений, в отличии от их файлов.

Для отображения информации о модуле e1000e, который является сетевым драйвером Intel PRO/1000, введите следующую команду, от имени пользователя root:

```
# modinfo e1000e
filename:       /lib/modules/3.10.0-
229.1.2.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko
version:        2.3.2-k
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
rhelversion:    7.1
srcversion:     0EED9FB230041A33B6B9BA9
alias:          pci:v00008086d000015A3sv*sd*bc*sc*i*
…
alias:          pci:v00008086d000010A4sv*sd*bc*sc*i*
alias:          pci:v00008086d0000105Fsv*sd*bc*sc*i*
alias:          pci:v00008086d0000105Esv*sd*bc*sc*i*
depends:        ptp
intree:         Y
vermagic:       3.10.0-229.1.2.el7.x86_64 SMP mod_unload modversions
signer:         RedOS Linux kernel signing key
sig_key:        34:B5:BC:A2:B7:06:D8:2E:72:A5:BE:3E:E4:09:BE:C7:19:5E:A5:08
sig_hashalgo:   sha256
parm:           debug:Debug level (0=none,...,16=all) (int)
```

```
parm:           copybreak:Maximum size of packet that is copied to a new buffer on
receive (uint)
parm:           TxIntDelay:Transmit Interrupt Delay (array of int)
parm:           TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:           RxIntDelay:Receive Interrupt Delay (array of int)
parm:           RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:           InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:           IntMode:Interrupt Mode (array of int)
parm:           SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:           KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:           WriteProtectNVM:Write-protect NVM [WARNING: disabling this can
lead to corrupted NVM] (array of int)
parm:           CrcStripping:Enable CRC Stripping, disable if your BMC needs the
CRC (array of int)
```

Описание основных полей вывода modinfo:

- **filename** – абсолютный путь к файлу объекта ядра **.ko**. Для вывода только этой информации, воспользуйтесь командой **modinfo -n**.
- **description** – короткое описание модуля. Для вывода только этой информации, воспользуйтесь командой **modinfo -d**.
- **alias** – поле **alias** представлено столько раз, сколько псевдонимов есть у модуля, если у модуля нет псевдонимов, после будет отсутствовать.
- **depends** – содержит разделенный запятыми список всех модулей, зависящих от текущего.

ПРИМЕЧАНИЕ

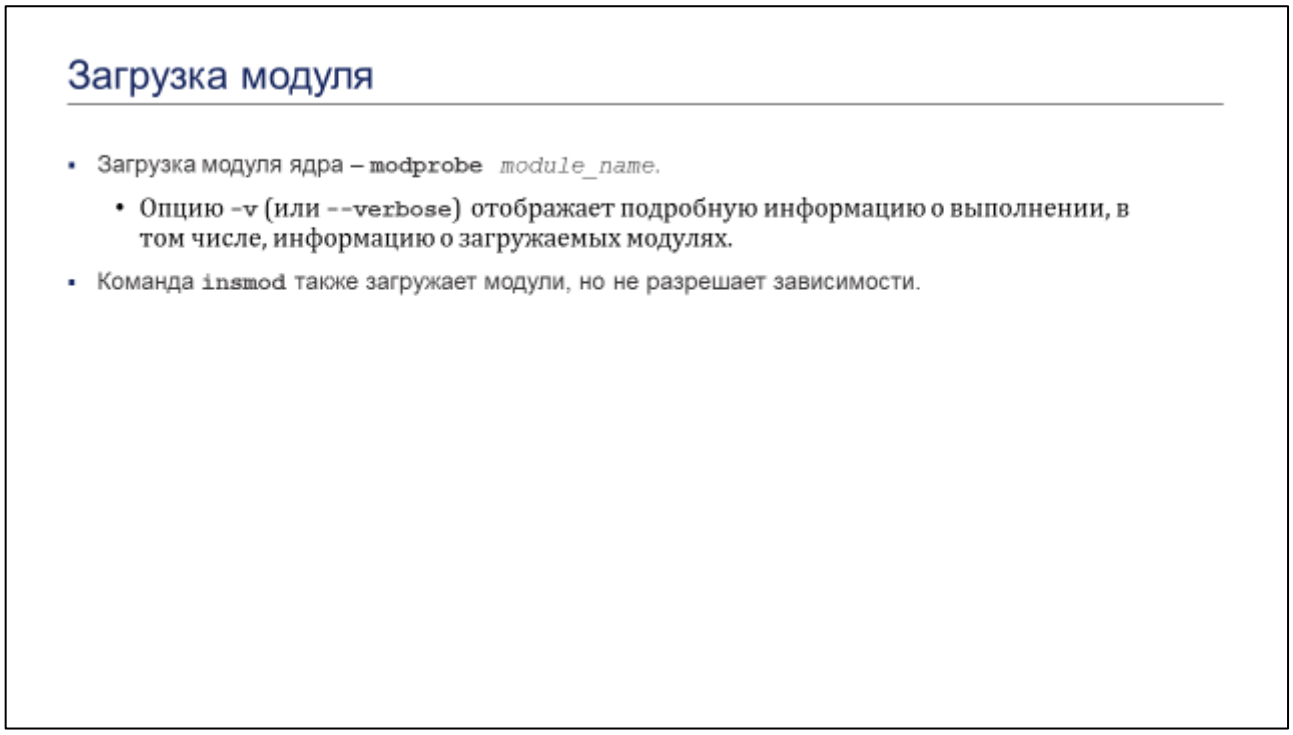
Если у модуля не зависимостей, поле **depends** может отсутствовать в выводе.

- **param** – каждое поле param представляет один параметр в следующей форме:

```
имя_параметра:описание
```

где:

- *имя_параметра* – это точный синтаксис, который можно использовать как параметр в командной строке или в строке опций в файле **.conf** в каталоге **/etc/modprobe.d/**.
- *описание* – краткое объяснение назначения параметра вместе с типом значения, которое может принимать параметр (такое как **int**, **unit** или **array of int**) в круглых скобках.



Для загрузки модуля ядра, выполните **modprobe** *имя_модуля*, от имени пользователя root. Например, для загрузки модуля **wacom**, воспользуйтесь следующей командой:

```
# modprobe wacom
```

По умолчанию, **modprobe** пытается загрузить модуль из **/lib/modules/***версия_ядра***/kernel/drivers/**. В этом каталоге, каждый тип модуля имеет собственный подкаталог, такой как **net/** и **scsi/**, для сети и интерфейсного драйвера SCSI соответственно.

Некоторые модули имеют зависимости – другие модули ядра, которые должны быть загружены до загрузки зависимого модуля. Команда **modprobe** всегда учитывает зависимости при выполнении операций. Когда вы запрашиваете **modprobe** загрузить указанный модуль ядра, в первую очередь она проверяет зависимости модуля и загружает их, в случае их наличия и, если они не загружены. **modprobe** проверяет зависимости рекурсивно: она загружает все зависимости зависимостей и так далее, чтобы убедится, что все зависимости загружены.

Вы можете использовать опцию **–v** (или **--verbose**), чтобы заставить **modprobe** отображать подробную информацию о выполнении, в том числе, информацию о загружаемых модулях.

Пример подробного вывода подгружаемых зависимостей для модуля Fibre Channel over Ethernet (**fcoe**):

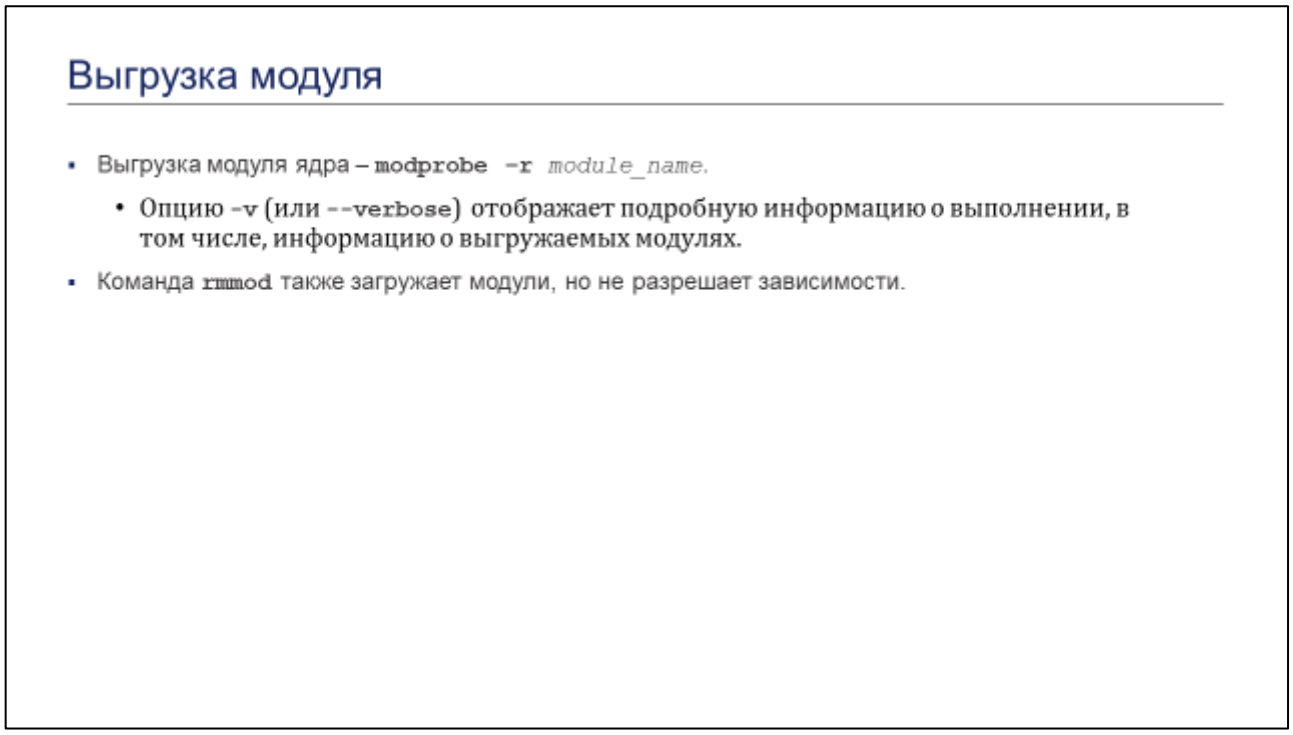
```
# modprobe -v fcoe
insmod /lib/modules/3.10.0-9.el7.x86_64/kernel/drivers/scsi/scsi_tgt.ko
insmod /lib/modules/3.10.0-9.el7.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko
insmod /lib/modules/3.10.0-9.el7.x86_64/kernel/drivers/scsi/libfc/libfc.ko
insmod /lib/modules/3.10.0-9.el7.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko
insmod /lib/modules/3.10.0-9.el7.x86_64/kernel/drivers/scsi/fcoe/fcoe.ko
```

В данном примере, вы можете увидеть, что modprobe загрузила модули **scsi_tgt**, **scsi_transport_fc**, **libfc** и **libfcoe** как зависимости перед загрузкой модуля **fcoe**. Также,

обратите внимание, что modprobe использует более примитивную команду **`insmod`** для вставки модулей в запущенное ядро.

ПРИМЕЧАНИЕ

Для загрузки модулей в запущенное ядро можно использовать команду **`insmod`**, но из-за того, что она не разрешает зависимости, рекомендуется использовать команду **`modprobe`**.



Вы можете выгрузить модуль ядра, при помощи команды **modprobe -r** *имя_модуля*, запущенной от имени пользователя root. Например, для выгрузки модуля wacom, воспользуйтесь следующей командой.

```
# modprobe -r wacom
```

Однако эта команда может завершится неудачей, если какой-либо процесс использует модуль **wacom** или в ядро загружен зависящий от него модуль.

Например, чтобы выгрузить модуль **firewire_ohci**, необходимо проследить всю цепочку его зависимостей:

```
# modinfo  -F depends firewire_ohci
firewire-core
# modinfo  -F depends firewire-core
crc-itu-t
# modinfo -F depends crc-itu-t
```

Как видно из примера кода выше, модуль **firewire_ohci** зависит от **firewire-core**, а тот в свою очередь зависит от **crc-itu-t**.

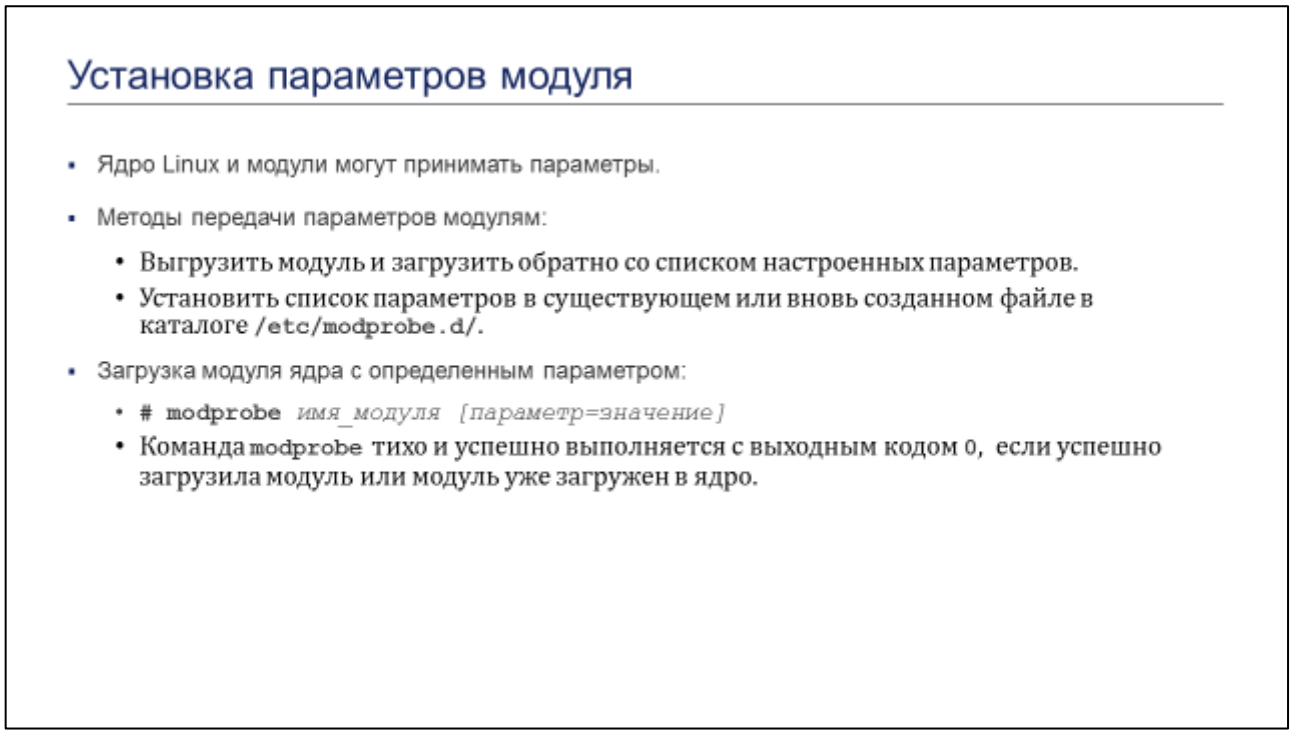
Вы можете выгрузить модуль при помощи команды **modprobe -v -r** *имя_модуля*, где **-r** сокращение **--remove**, а **-v** сокращение **--verbose**:

```
# modprobe -r -v firewire_ohci
rmmod firewire_ohci
rmmod firewire_core
rmmod crc_itu_t
```

Вывод показывает, что модули выгружены в обратном порядке, от того, как они были загружены, данный вывод говорит о том, что никакие процессы, зависящие от этих модулей, не были выгружены.

ПРИМЕЧАНИЕ

Для выгрузки модулей из запущенного ядра можно использовать команду **`rmmod`**, но из-за того, что она выгружает только указанный модуль, рекомендуется использовать команду **`modprobe`**.



Как и само ядро, модули могут принимать параметры, которые меняют их поведение. В большинстве случае, параметры по умолчанию отлично подходят, но иногда необходимо или желательно установить параметр модуля, запущенного в ядро в иное значение. Существует два метода сделать это:

1. Вы можете выгрузить модуль вместе с зависимостями, при помощи команды **modprobe -r** и затем загрузить обратно при помощи **modprobe** вместе со списком настроенных параметров. Этот метод часто используется, когда у модуля не много зависимостей или тестируются различные комбинации параметров без установки их на постоянно основе.
2. В качестве альтернативы, можно установить список параметров в существующем или вновь созданном файле в каталоге **/etc/modprobe.d/**. Этот метод делает параметры постоянными, они будут использоваться при загрузке системы и вызове команды **modprobe**.

Для загрузки модуля ядра с определенным параметром, используйте команду **modprobe** в следующем формате:

```
# modprobe имя_модуля [параметр=значение]
```

Когда вы загружаете модуль с определенными параметрами в командной строке:

- Вы можете ввести несколько параметров и значений, разделив их пробелами.
- Некоторые параметры модулей используют список, разделенный запятыми, в качестве значения. При вводе такого списка нельзя использовать пробелы, иначе **modprobe** некорректно интерпретирует значение после пробела, как дополнительный параметр.
- Команда **modprobe** тихо и успешно выполняется с выходным кодом **0**, если успешно загрузила модуль или модуль уже загружен в ядро.

Таким образом, вы должны предварительно убедиться, что модуль еще не загружен, прежде чем загружать его с настроенными параметрами. Команда **modprobe** не перезагружает модуль автоматически и не выдает предупреждения о том, что он уже загружен.

Рассмотрим рекомендованные шаги для установки настраиваемого параметра и загрузки модуля ядра. В качестве примера, возьмем модуль **e1000e**, который является драйвером сетевого адаптера Intel PRO/1000.

1. Убедитесь, что модуль еще не загружен в ядро:

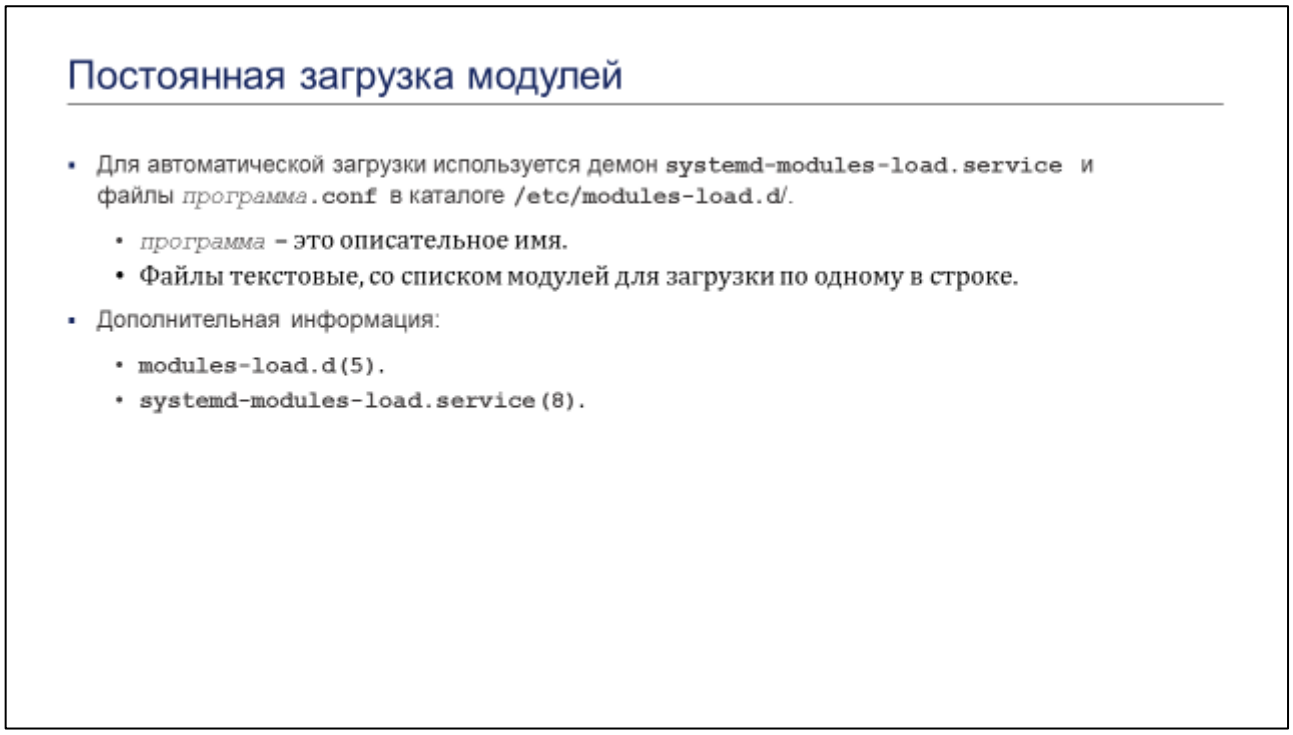
```
# lsmod | grep e1000e
```

Если вывод команды покажет загруженный модуль, его необходимо выгрузить.

2. Загрузите модуль, перечислив все необходимые настраиваемые параметры после имени модуля. Например, если вы хотите загрузить драйвер сетевого адаптера Intel PRO/1000 с уровнем частоты прерывания, установленным в **3000** прерываний в секунду, для первого, второго и третьего интерфейсов драйвера и включить отладку, введите следующую команду от имени пользователя root:

```
# modprobe e1000e InterruptThrottleRate=3000,3000,3000 debug=1
```

Данный пример иллюстрирует передачу нескольких значений, разделенных запятыми, в одном параметре без использования пробелов.



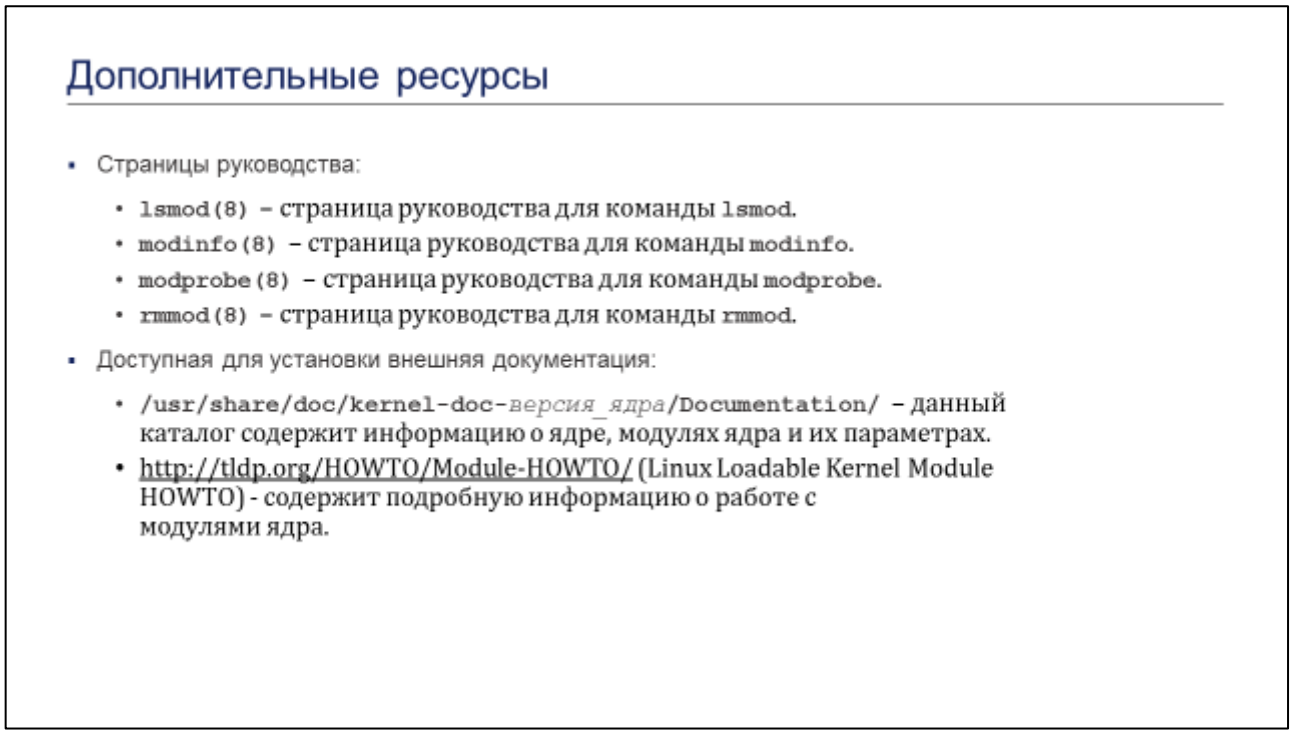
Как можно было заметить ранее, множество модулей ядра загружаются автоматически в процессе загрузки системы. Вы можете указать дополнительные модули для автоматической загрузки демоном **systemd-modules-load.service** за счет создания файла *программа***.conf** в каталоге **/etc/modules-load.d/**, где *программа* – это любое описательное имя на ваш выбор. Файлы в каталоге **/etc/modules-load.d/** текстовые, со списком модулей для загрузки по одному в строке.

Пример текстового файла для загрузки модуля **virtio-net.ko** (Сетевой драйвер Virtio).

Чтобы создать файл для загрузки модуля **virtio-net.ko**, создайте файл **/etc/modules-load.d/virtio-net.conf** со следующем содержимым:

```
# Load virtio-net.ko at boot
virtio-net
```

Для получения дополнительной информации обратитесь к страницам руководства **modules-load.d(5)** и **systemd-modules-load.service(8)**.



Для получения дополнительной информации по модулям ядра и их утилитам, смотрите следующие ресурсы:

- Страницы руководства:
  - **lsmod(8)** – страница руководства для команды **lsmod**.
  - **modinfo(8)** – страница руководства для команды **modinfo**.
  - **modprobe(8)** – страница руководства для команды **modprobe**.
  - **rmmod(8)** – страница руководства для команды **rmmod**.
- Доступная для установки внешняя документация:
  - **/usr/share/doc/kernel-doc-***версия_ядра***/Documentation/** - данный каталог, который предоставляется пакетом **kernel-doc**, содержит информацию о ядре, модулях ядра и их параметрах. Для установки документации, воспользуйтесь следующей командой от имени пользователя root:

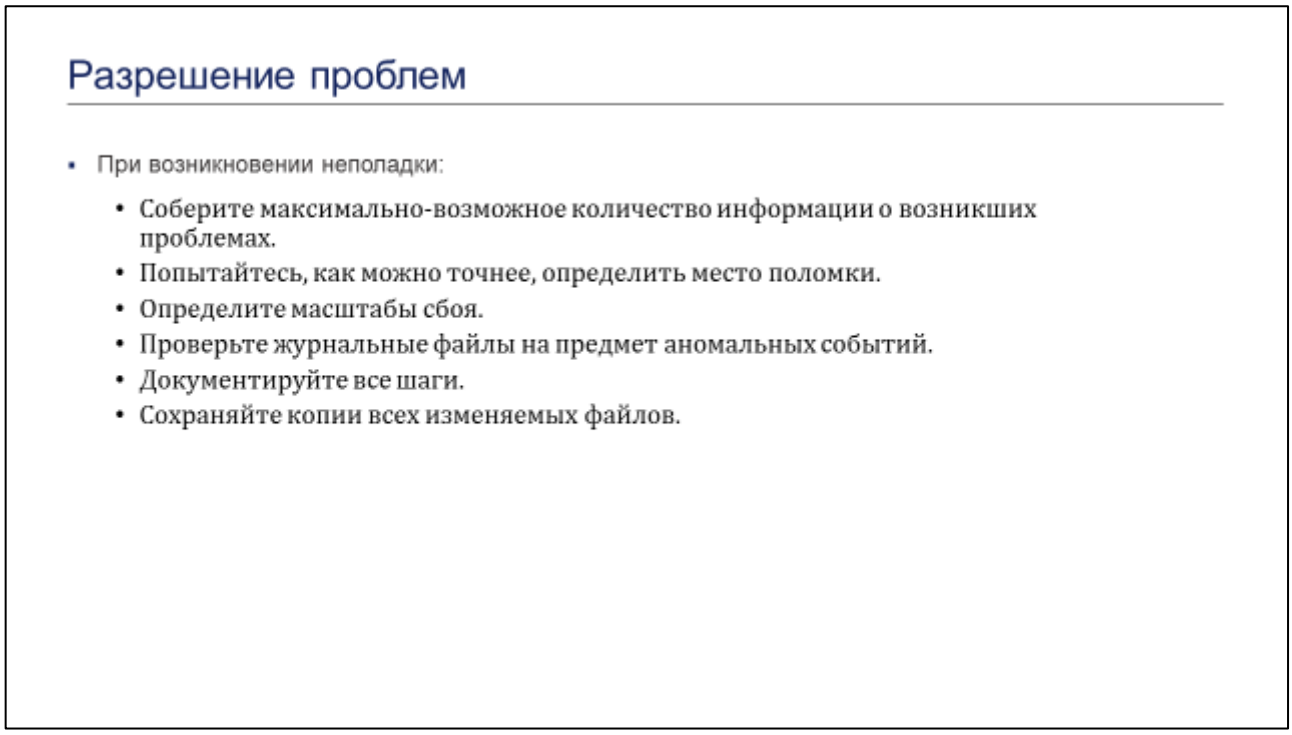
```
# yum install kernel-doc
```

  - Linux Loadable Kernel Module HOWTO (http://tldp.org/HOWTO/Module-HOWTO/) – содержит подробную информацию о работе с модулями ядра.

Модуль 9. Устранение неисправностей РЕД ОС

В этом модуле рассматриваются:

- Подходы к устранению неисправностей
- Устранение неисправностей загрузки системы
- Устранение неисправностей накопителей и файловых систем
- Устранение неисправностей сети

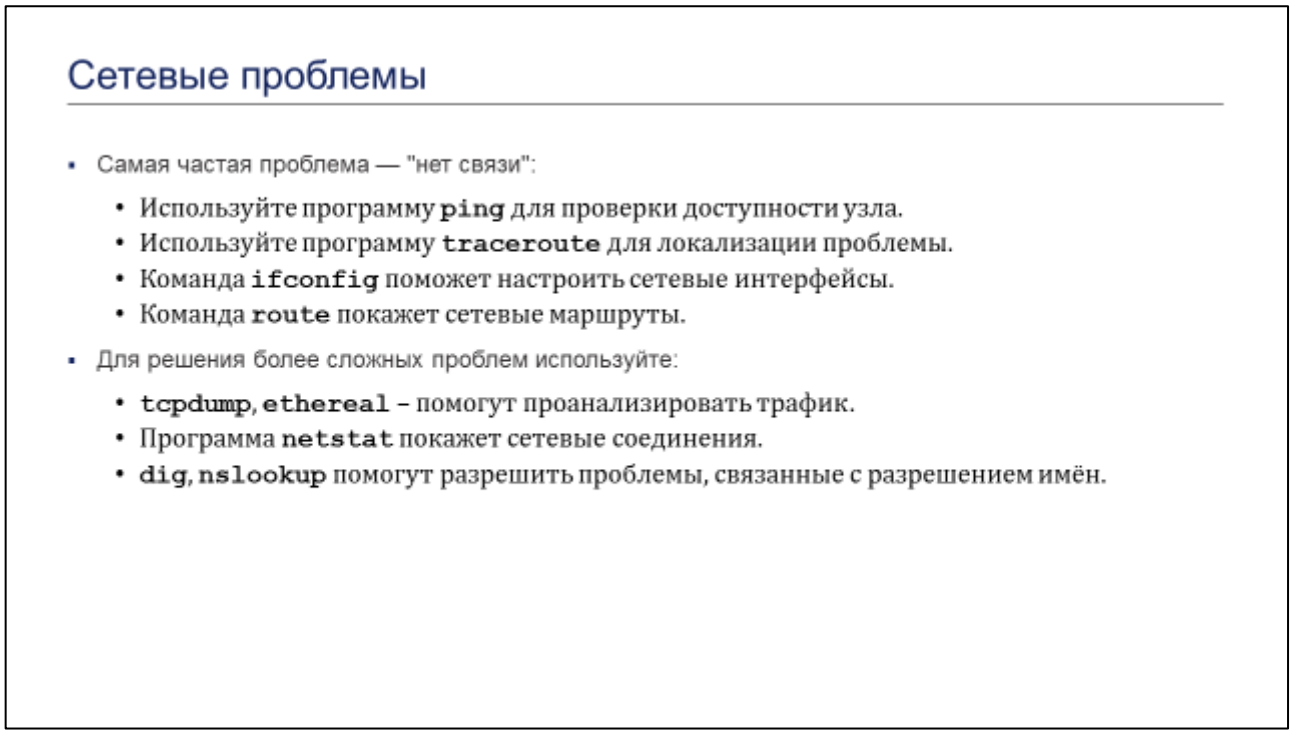


Linux — очень сложная система и подвергается правилу "надёжность системы зависит от её сложности", поэтому иногда случаются сбои в работе. Сбои бывают разные и приводят к разным результатам, начиная от неудобства работы охранника и заканчивая остановкой деятельности всего предприятия. Естественно, что чем грамотнее вы строили систему, тем она будет менее подвержена крупным сбоям, иначе вы рискуете получить риск масштабного краха даже при возникновении незначительных проблем в работе системы.

Устранение и поиск неполадок — искусство. Чтобы найти проблему, требуется не только хорошее понимание процессов, происходящих в системе, но и отличная смекалка и везение. Сложность поиска неисправности усугубляется тем, что система может не выполнять своих функций, а следовательно, что-то в организации не будет работать и вы получите начальника, стоящего в дверях с суровым выражением лица. Время — главный враг системного администратора, при устранении неполадок. Всегда старайтесь досконально разобраться в проблеме, но если организации грозят большие убытки, то постарайтесь в минимальные сроки восстановить работоспособность, пусть это будет временное и некрасивое решение, главное дать людям возможность работать, когда вы это сделаете, тогда появится время на изучение проблемы и более красивого её решения.

Проблемы могут носить самый разнообразный характер, это может быть и неправильная конфигурация, и сбой аппаратной части, это может быть повреждённая файловая система, как следствие удара молнии. Всё что угодно, администратор должен быть готов к самым большим неожиданностям. Иногда проблема бывает вызвана ошибкой в программном обеспечении, такие ошибки лучше предупреждать путём своевременного обновления системы до актуального состояния.

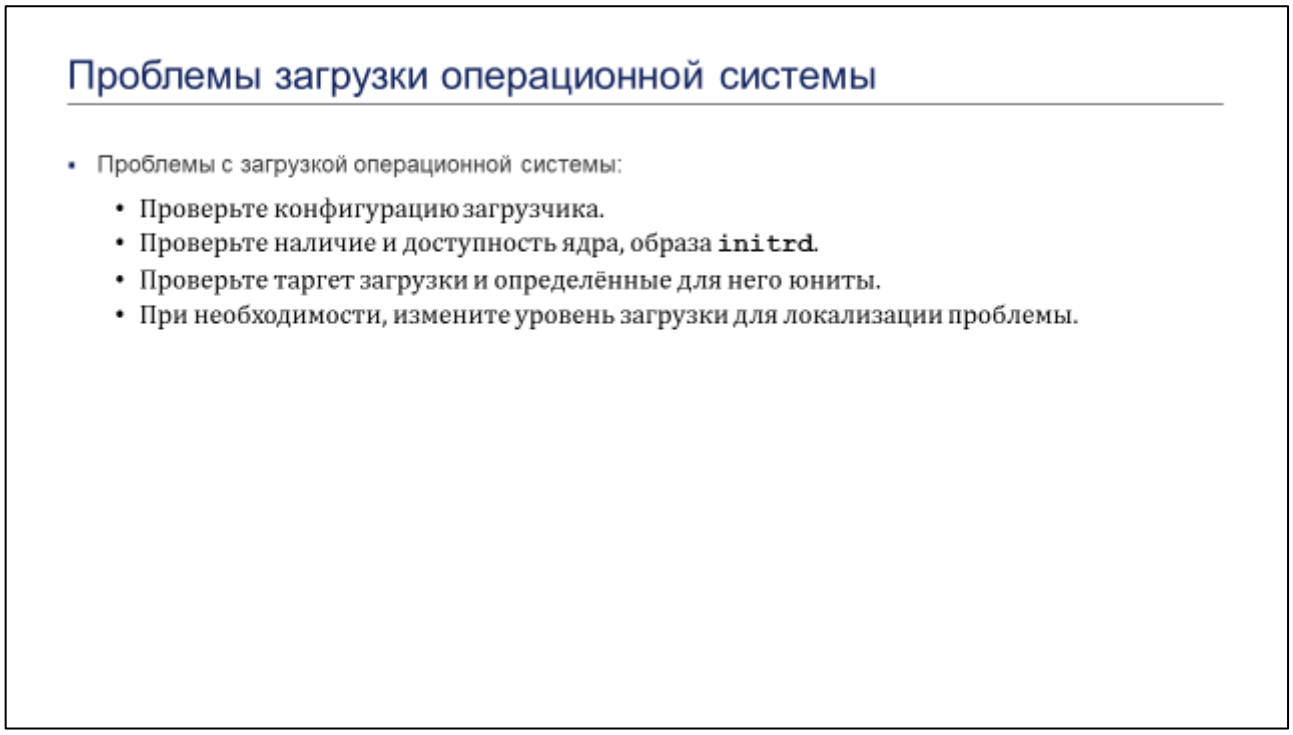
Никогда не забывайте о том, что все проблемы решаемы, но так утверждать может только



Являетесь вы системным или сетевым администратором, вам всё равно придётся решать проблемы, связанные с функционированием сети. При поступлении заявки с текстом "не открывается сайт", сразу связывайтесь с автором и просите пояснить, что пишет web-браузер, в кодах ошибок очень многое говорится, ведь проблема может быть и в разрешении имён и в вашем маршрутизаторе, а может сервер, на котором находится сайт, и вовсе выключили.

Доступность удалённого узла вы всегда можете проверить программой ping, но всегда помните, как она работает. Правильно интерпретировать результаты работы программы вы сможете только тогда, когда поймёте, что делает эта программа. Если при отправке ICMP-пакета программой ping вы не получили ответного пакета, то это ни в коем случае не даёт 100% гаратнию, что узел не доступен. Это может говорить о том, что сетевой пакет отфильтровался на каком-то этапе, ведь сетевые брандмауэры могут фильтровать icmp-трафик, но пропускать весь остальной.

Случается, что проблемы не столько просты. В больших сетях иногда случаются "сетевые штормы", когда через сетевые устройства проходит такое количество трафика, что они становятся перегруженными, замедляется работа сети. В таком случае, команда ping поможет вам, разве что, убедиться в том, что в сети неполадки. Вам необходимо использовать анализаторы трафика и посмотреть, какие пакеты, откуда и куда они идут, чтобы локализовать проблему и перекрыть её источник.



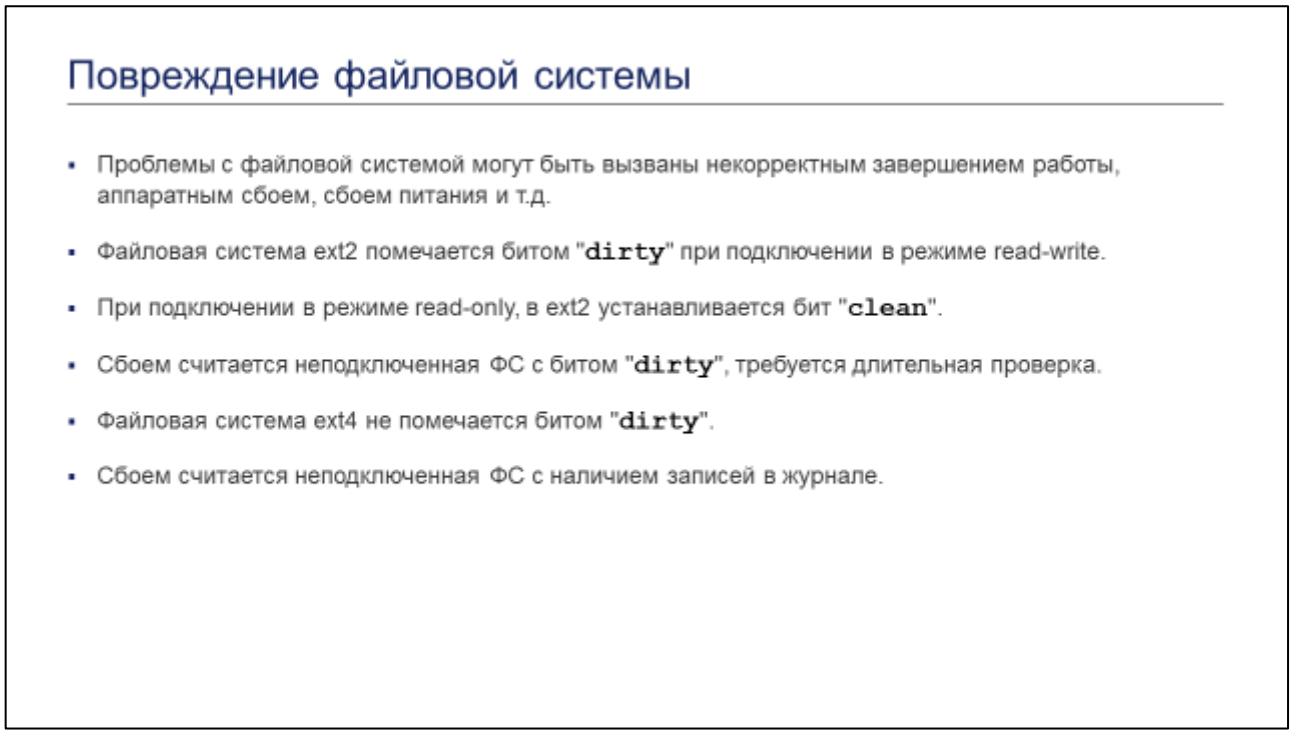
Проблемы, связанные с загрузкой системы, являются одними из самых сложных, так как сервер может не перезагружаться месяцами и действие, из-за которого появился сбой, могло быть выполнено в любой из дней работы сервера, а проблема проявится только при перезагрузке. Именно для этого необходимо документирование всех изменений, вносимых в систему.

В процессе решения проблем, связанных с загрузкой системы, очень важно понимать и помнить весь процесс запуска, чтобы быстро определить неисправный элемент.

Если загрузка оборвалась на этапе загрузки ядра, то проверьте сначала конфигурацию загрузчика GRUB, убедитесь, что в ней нет опечаток, что присутствуют обязательные параметры и они указывают на валидные объекты. Убедитесь в наличии и доступности ядра, а также образа **`initrd`**, отсутствие которого может привести к сбою загрузки.

Если загрузка была прервана после загрузки ядра, при вызове процесса **`init`**, то начинайте проверять его элементы, начните с файла **`/etc/inittab`**. Не исключено, что проблема была вызвана повреждённой файловой системой. Если вы загружаетесь в 5-м уровне запуска, и после загрузки всех элементов вы не можете попасть в систему, то обратите внимание на X-сервер, возможно проблема скрыта в нём.

Всегда обращайте пристальное внимание на сообщения, выдаваемые системой, в них может быть скрыта вся необходимая, для решения проблемы, информация.

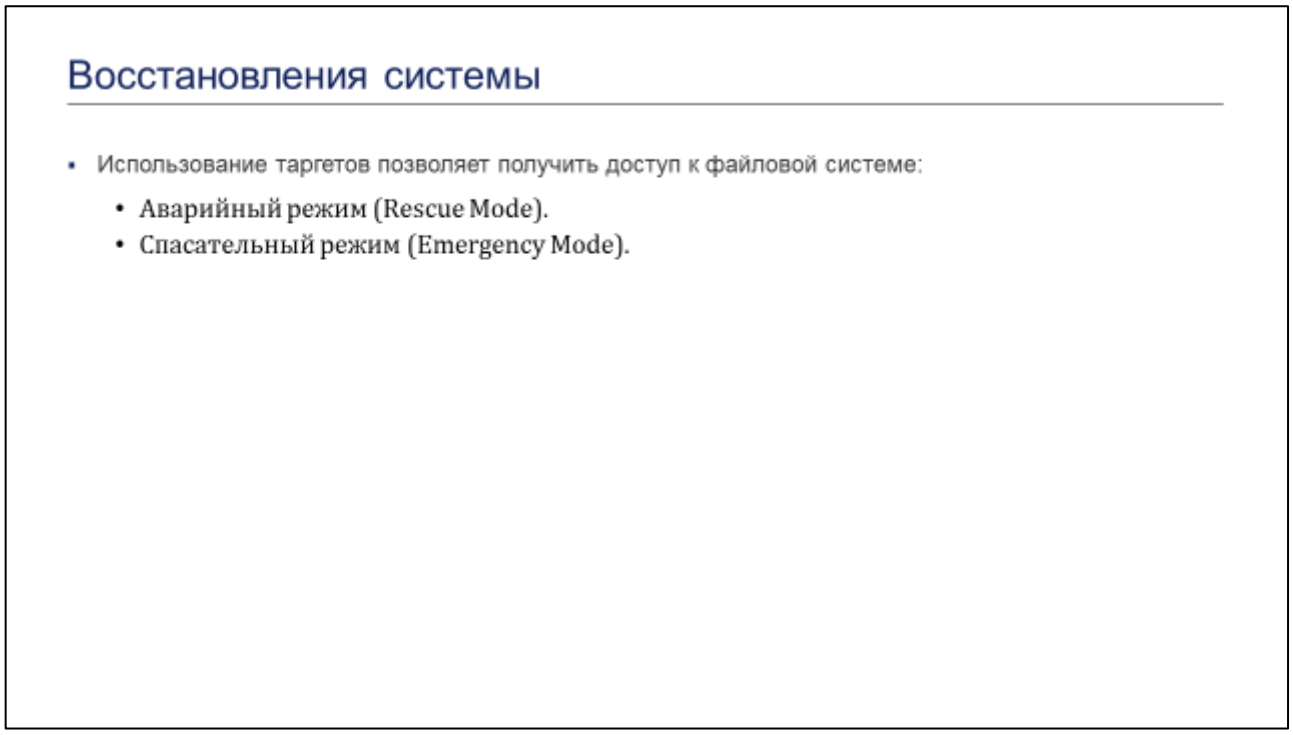


Повреждения файловой системы случаются очень часто, так как бывают вызваны крахом системы, либо сбоем питания, либо иными событиями, часто происходящими в процессе работы. Такого рода повреждения связаны с тем, что запись на диск производится не сразу, а постепенно. Сначала данные помещаются в буфер, после чего постепенно записываются на диск, естественно, что при исчезновении питания, содержимое этих буферов не сохраняется.

При подключении файловой системы ext2 она помечается битом "dirty", который будет снят при корректном отключении файловой системы, которое гарантирует синхронизацию данных между буфером и диском. Когда файловая система подключается и обнаруживается установленный бит "dirty", это означает, что файловая система была некорректно отключена и может иметь повреждённую информацию. В таком случае требуется длительная проверка данных.

Файловая система ext3 отличается от ext2 только наличием журнала, соответственно, при подключении такой файловой системы и обнаружении в журнале записей, делается заключение, что система повреждена. Но поскольку в журнале находится информация о заданиях, необходимых для выполнения, то процесс может быть продолжен.

В случае повреждения файловой системы, попробуйте восстановить её с помощью программы **`fsck`**.



Аварийный режим (Rescue Mode) – предоставляет удобный однопользовательское окружение, которое позволит вам восстановить вашу систему в случае невозможности загрузиться в нормальное состояние. В аварийном режиме система пытается смонтировать все локальные файловые системы и запускает только важные системные сервисы, но не запускает сетевые интерфейсы и не позволяет параллельный вход в систему. В RedOS 7 аварийный режим полностью соответствует однопользовательскому режиму (Single User Mode) и требует ввода пароля пользователя root.

Чтобы сменить текущий таргет и войти в аварийный режим в текущей сессии, введите следующую команду от имени пользователя root:

```
# systemctl rescue
```

Эта команда похожа на **`systemctl isolate rescue.target`**, но она также посылает информационное сообщение всем пользователям, которые на данный момент находятся в системе. Чтобы предотвратить отправку сообщения, введите команду с указанием опции **`--no-wall`**:

```
# systemctl --no-well rescue
```

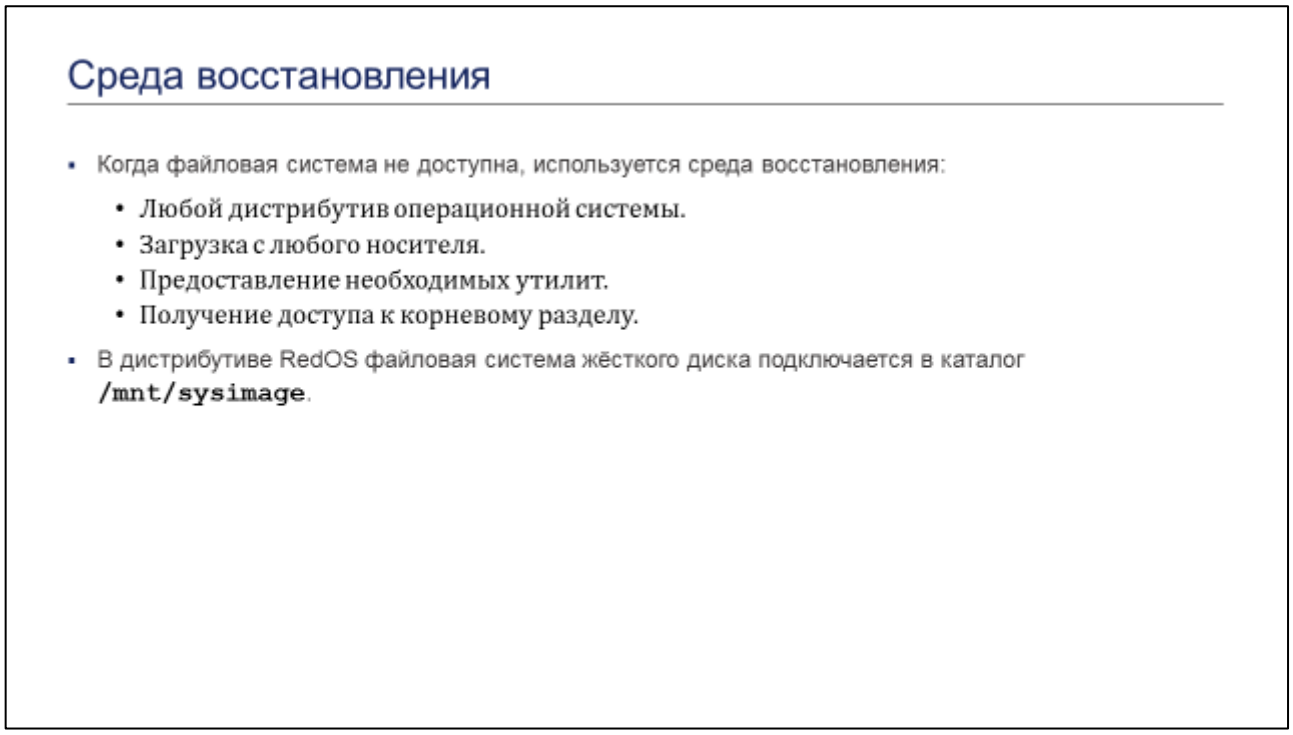
Спасательный режим (Emergency Mode) – предоставляет наиболее минимальное окружение из возможных и позволяет вам восстанавливать вашу систему даже в ситуации, когда система не может войти в аварийное окружение (Rescue Mode). В спасательном окружении система монтирует корневую файловую систему только для чтения, не пробует монтировать остальные локальные файловые системы, не активирует сетевые интерфейсы и запускает только несколько ключевых сервисов. В RedOS 7 спасательный режим требует пароль пользователя root.

Для смены текщего таргета и входа в спасательный режим, введите следующую команду от имени пользователя root:

```
# systemctl emergency
```

Эта команда похожа на **systemctl isolate emergency.target**, но она также посылает информационное сообщение всем пользователям, которые на данный момент находятся в системе. Чтобы предотвратить отправку сообщения, введите команду с указанием опции **--no-wall**:

```
# systemctl --no-well emergency
```



Среда восстановления используется в случае, когда загрузка с жёсткого диска невозможна, по какой-либо причине. При загрузке среды восстановления, она пытается обнаружить установленные копии дистрибутива и предлагает администратору выбрать, с какой из копий необходимо работать.

Найденный корневой раздел установленной ОС будет подключен в каталог **/mnt/sysimage**. Естетственно, если файловая система повреждена, то среда восстановления может оказаться бессильной, при переключении на вторую виртуальную консоль вы получите доступ к программе **fdisk**.

Чтобы загрузиться в режим восстановления, вставьте установочный компакт-диск и при загрузке, в качестве параметра, укажите **rescue**:

```
linux rescue
```

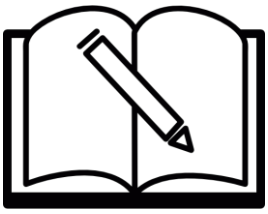
Модуль 10. Автоматизация установки РЕД ОС

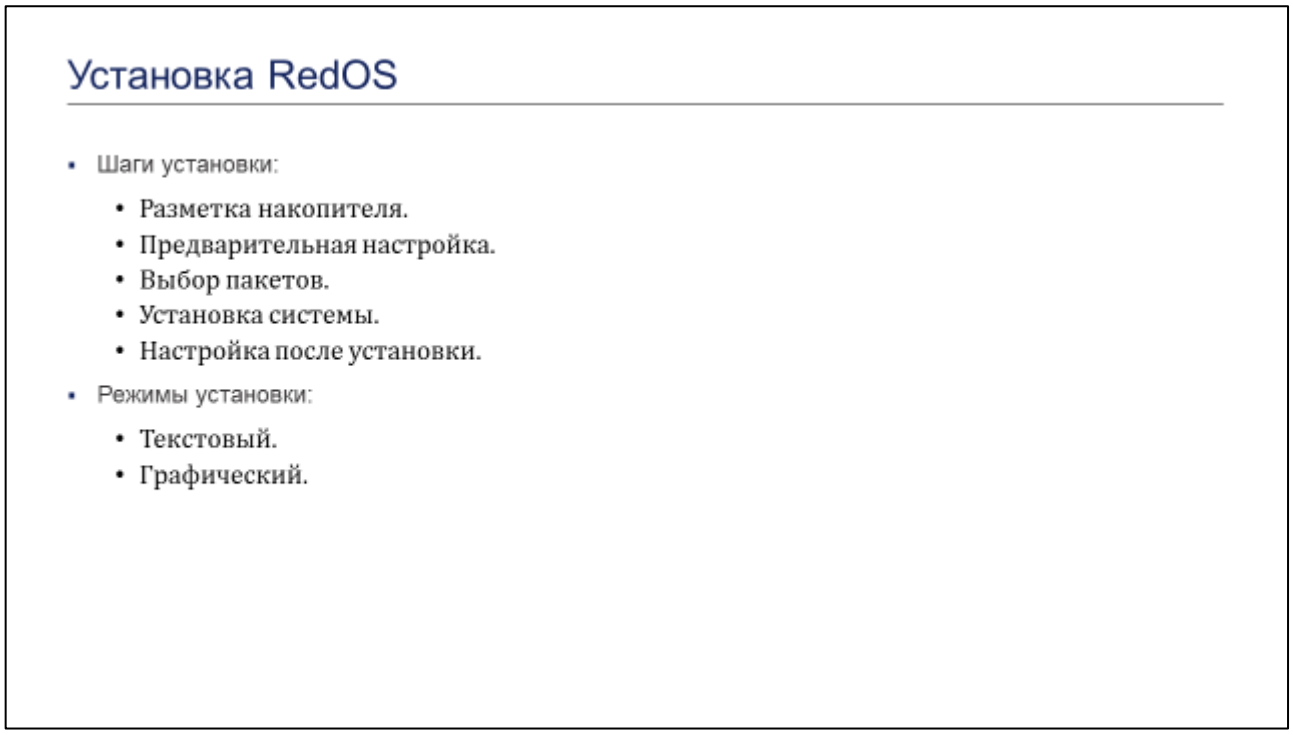
В этом модуле рассматриваются:

Знакомство с возможностями Anaconda

Создание файла автоматических ответов

Автоматическая установка РЕД ОС





Перед установкой убедитесь, что всё оборудование, необходимое для работы, поддерживается ядром операционной системы.

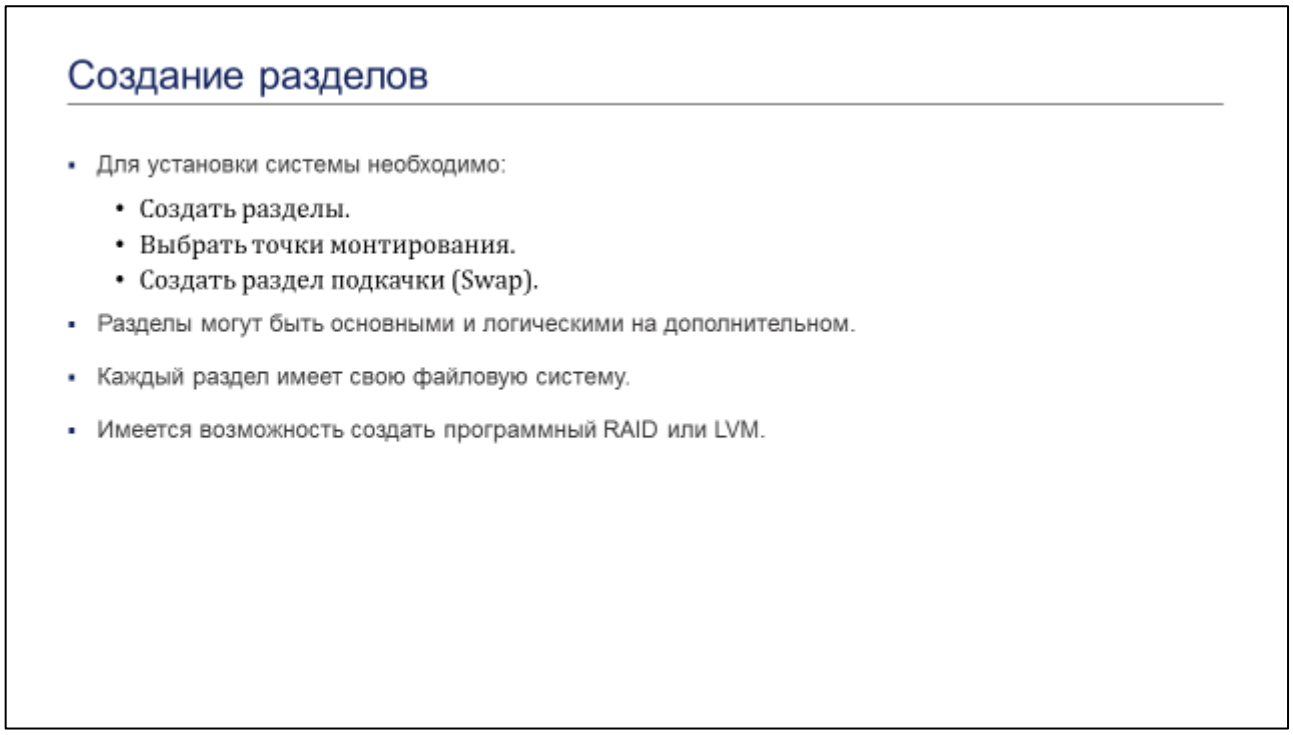
Установить дистрибутив операционной системы, построенной на ядре Linux, очень просто. На сегодняшний день, практически все программы установки предлагают пользователям, помимо классического – текстового, графический установщик. Кроме упомянутых двух типов установки, существует ещё один – сетевая установка, интересна администраторам, под управлением которых находятся удалённые компьютеры, либо при потребности установки на большое число систем.

Чтобы приступить к установке вставьте первый загрузочный диск в привод. Убедитесь, что в BIOS системы указан верный приоритет загрузочных устройств, первым должен стоять привод компакт-дисков. У каждого дистрибутива оригинальное приглашения к началу инсталляции, RedOS предлагает небольшое меню с описанием каждого типа. Если вы желаете начать установку в текстовом режиме, то необходимо добавить к командной строке:

```
… text
```

Если же вы хотите начать установку в графическом режиме, то просто нажмите клавишу <Enter>.

Система установки Anaconda задаст ряд вопросов, касающихся раскладки клавиатуры, даты и времени и т.д. Важной составляющей процесса установки является создание разделов на жёстком диске.



Физические диски делятся на разделы, которые могут содержать либо файловую систему, либо выделяться в качестве подкачки, как составляющая виртуальной памяти. Создание разделов позволяет изолировать данные друг от друга, это может быть множество операционных систем, либо домашние каталоги пользователей. Первые четыре раздела являются первичными разделами. При создании более чем 4-х разделов, выделяется специальный extended раздел, в котором находятся логические разделы.

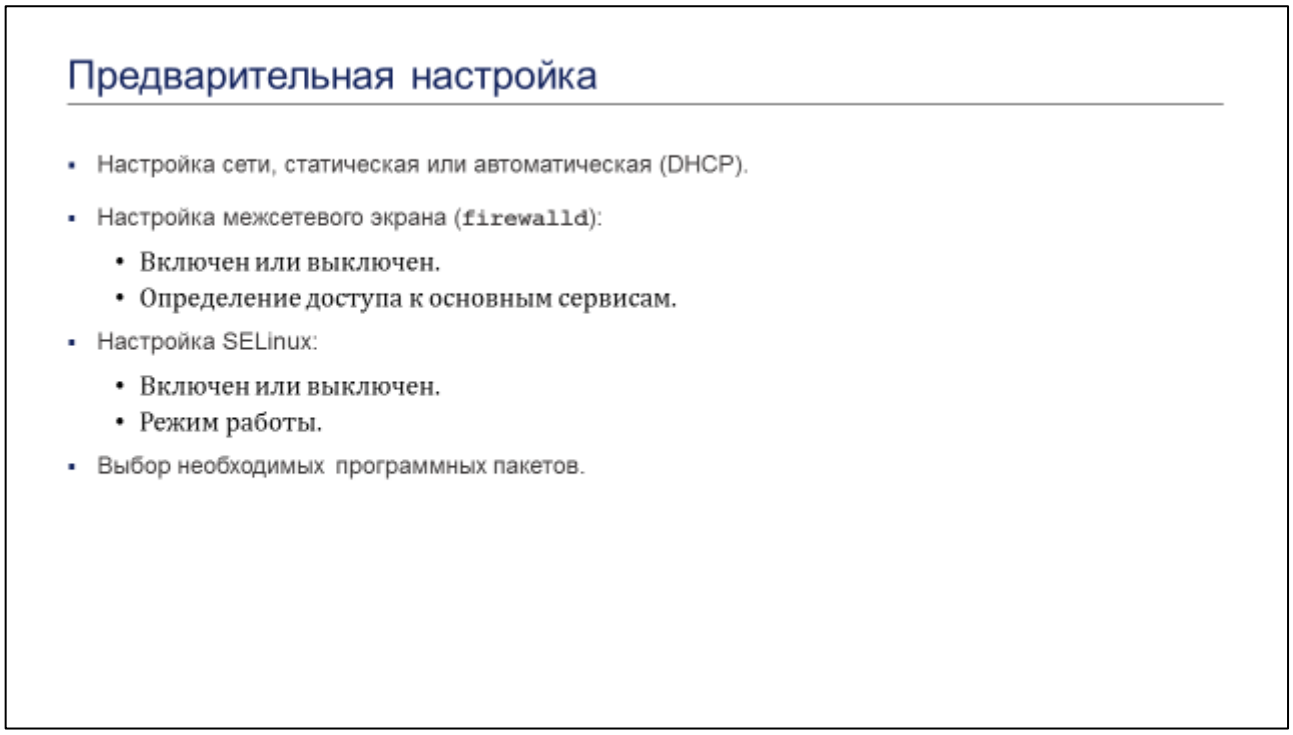
Разделы могут быть объединены в некую сущность, массив. Такое объединение осуществляется с помощью аппаратного RAID-контроллера, либо его программного аналога. Существует несколько уровней RAID-массивов. Массив уровня RAID0 предназначен для обеспечения отказоустойчивости, когда все данные, записываемые на один раздел, будут отражаться на втором "зеркальном" разделе. Массив уровня RAID1 предназначен для увеличения скорости операций записи на диск, запись будет производиться поочерёдно, сначала на один, а затем на другой раздел, входящий в массив. При этом размер массива уровня RAID1 будет равен суммарному размеру разделов, входящих в него.

Помимо RAID-массивов существует система LVM (Logical Volume Manager), которая объединяет несколько разделов в один с целью создания логической группы, в которой администратор может создавать логические тома. Использование LVM делает возможным выполнения таких операций, как изменение размера тома, создания снимков. Далее мы более подробно рассмотрим обе технологии. Графический режим установки системы позволяет администратору не только создавать простые разделы, но и организовать программный RAID-массив или LVM-группы.

Для создания раздела необходимо выбрать физическое устройство и нажать кнопку создания нового раздела. Установщик предложит указать следующие параметры:

- Точка монтирования. Каталог, в иерархии файловой системы, к которому будет подключен раздел, достаточно указать корневой раздел - **/**.

- Тип файловой системы. По умолчанию используется файловая система **`xfs`**. Помните, что у раздела swap нет точки монтирования, чтобы его создать, необходимо выбрать в качестве файловой системы – **`swap`**.
- Размер файловой системы.



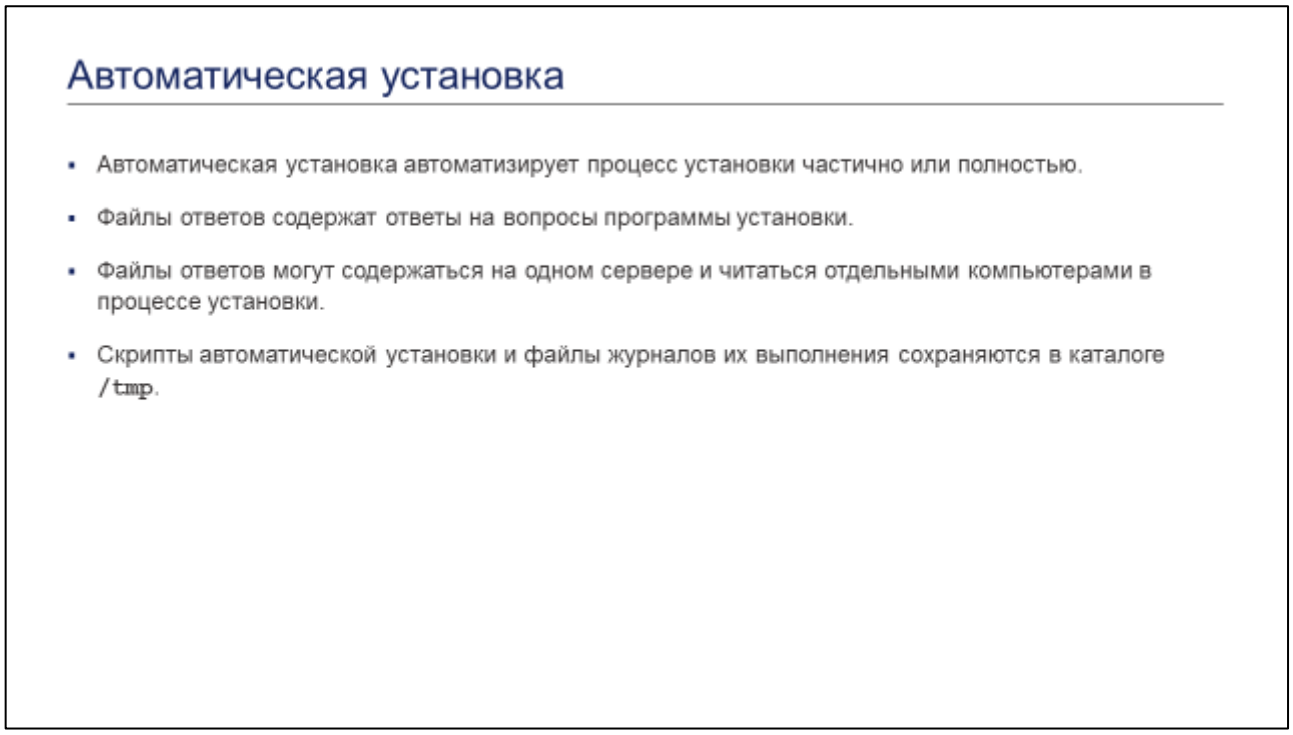
Вы можете при установке указать сетевые реквизиты, либо указать на использование DHCP для автоматического получения сетевых реквизитов.

Программа установки предложит включить или выключить системный брандмауэр. Если пользователь указал на включение, то будет предложено указать минимальные настройки:

- Разрешить весь сетевой трафик для определённого интерфейса.
- Разрешить входящий трафик для определённых сервисов.

SELinux – Secure Enhanced Linux, система управления доступом, основанная на концепции MAC (Mandatory Access Control). В отличие от традиционной дискреционной системы контроля доступа, когда пользователь определяет уровень доступа к объекту, SELinux обеспечивает контроль на уровне системных политик, которые описывают, кто и какие действия может совершать над тем или иным объектом. Установщик предлагает на выбор три режима работы SELinux:

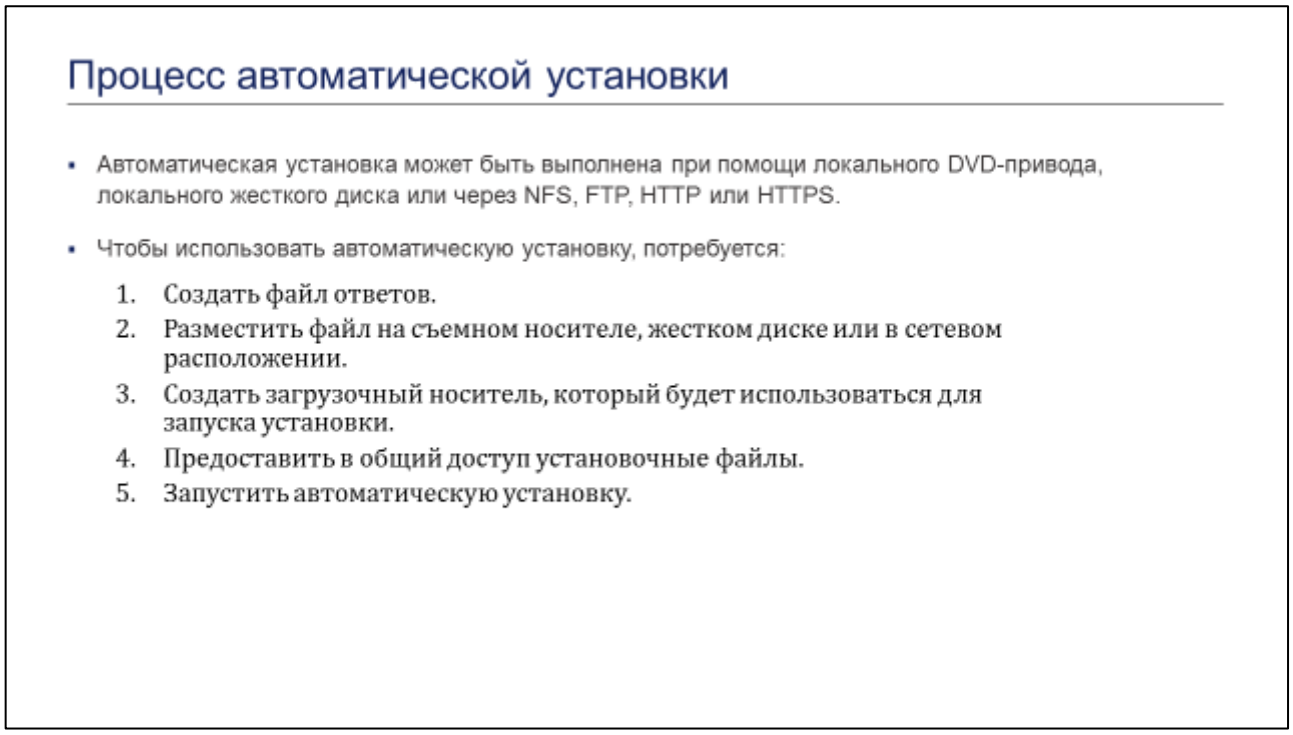
- **`Disabled`** – отключен.
- **`Permissive`** – включён, но производимые ограничения только регистрируются в журнальных файлах, не внося реальных ограничений в действия пользователей.
- **`Enforcing`** – SELinux полностью включён и применяет системные политики по их прямому назначению.



Автоматическая установка (Kickstart Installation) автоматизирует процесс установки частично или полностью. Файлы автоматической установки содержат ответы на все вопросы, которые задает программа установки, такие как выбор временной зоны, разметка разделов или выбор пакетов для установки. Предоставление подготовленного файла ответов в момент запуска установки, позволяет выполнить установку без дальнейшего участия пользователя. Это особенно полезно, если необходимо выполнить одновременную установку CentOS (RHEL) на множестве систем.

Файлы автоматической установки могут содержаться на одном сервере и читаться отдельными компьютерами в процессе установки.

Все скрипты автоматической установки и файлы журналов их выполнения сохраняются в каталоге **/tmp**, что помогает отлаживать ошибки установки.

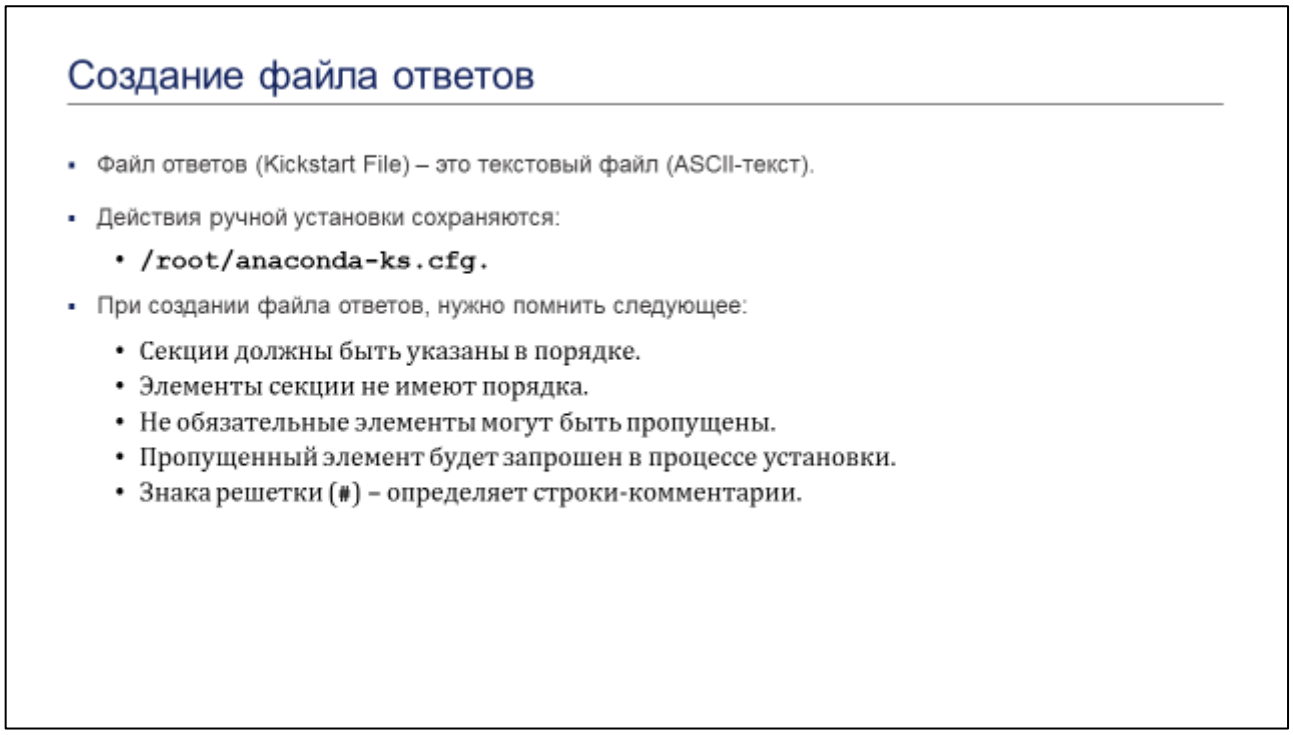


Процесс выполнения автоматической установки (Kickstart Installation).

Автоматическая установка может быть выполнена при помощи локального DVD-привода, локального жесткого диска или через NFS, FTP, HTTP или HTTPS.

Чтобы использовать автоматическую установку, потребуется:

1. Создать файл ответов.
2. Разместить файл на съемном носителе, жестком диске или в сетевом расположении.
3. Создать загрузочный носитель, который будет использоваться для запуска установки.
4. Предоставить в общий доступ установочные файлы.
5. Запустить автоматическую установку.



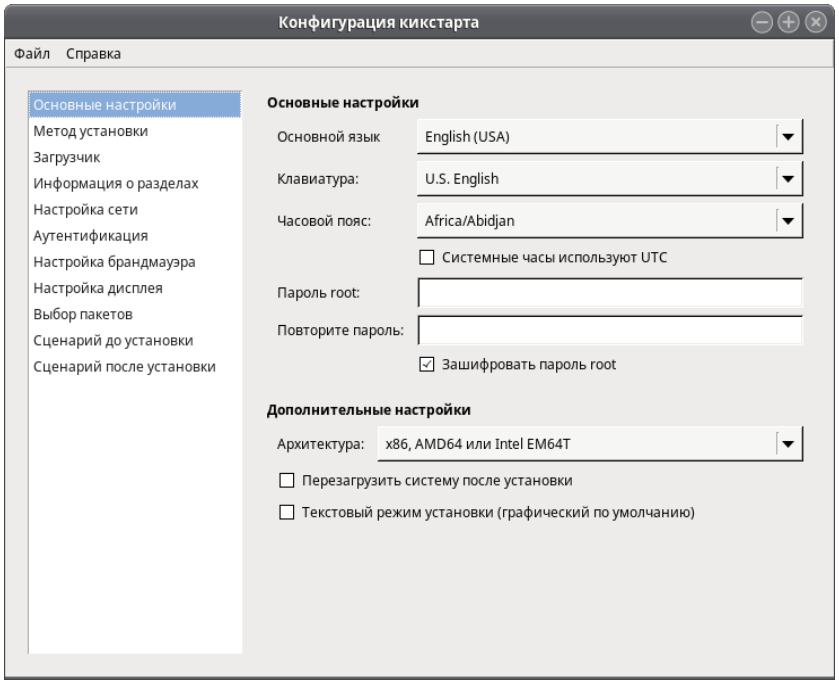
Файл ответов (Kickstart File) – это текстовый файл, содержащий поддерживаемые ключевые слова, которые используются в качестве директив в процессе установки. Любой текстовый редактор, способный сохранить файлы с ASCII-текстом, такой как Gedit или Vim в Linux или Блокнот (Notepad) в Windows, может быть использован для создания файла ответов (Kickstart File). Имя файла, который будет содержать ответы для программы установки не имеет значения, однако рекомендуется использовать простое имя, так как вам придется использовать его позже в конфигурационных файлах или диалоговых окнах.

Рекомендованный подход к созданию файла ответов (Kickstart File) – это выполнить установку вручную на одной из систем. После завершения установки, все выбранные в процессе установки действия будут сохранены в файле **anaconda-ks.cfg**, который располагается в каталоге /root/ установленной системы. Вы можете скопировать файл, внести в него любые изменения и использовать для дальнейших установок.

Инструмент Kiskstart Configurator (system-config-kickstart), который позволяет создать файл ответов (Kickstart File) при помощи графического мастера.

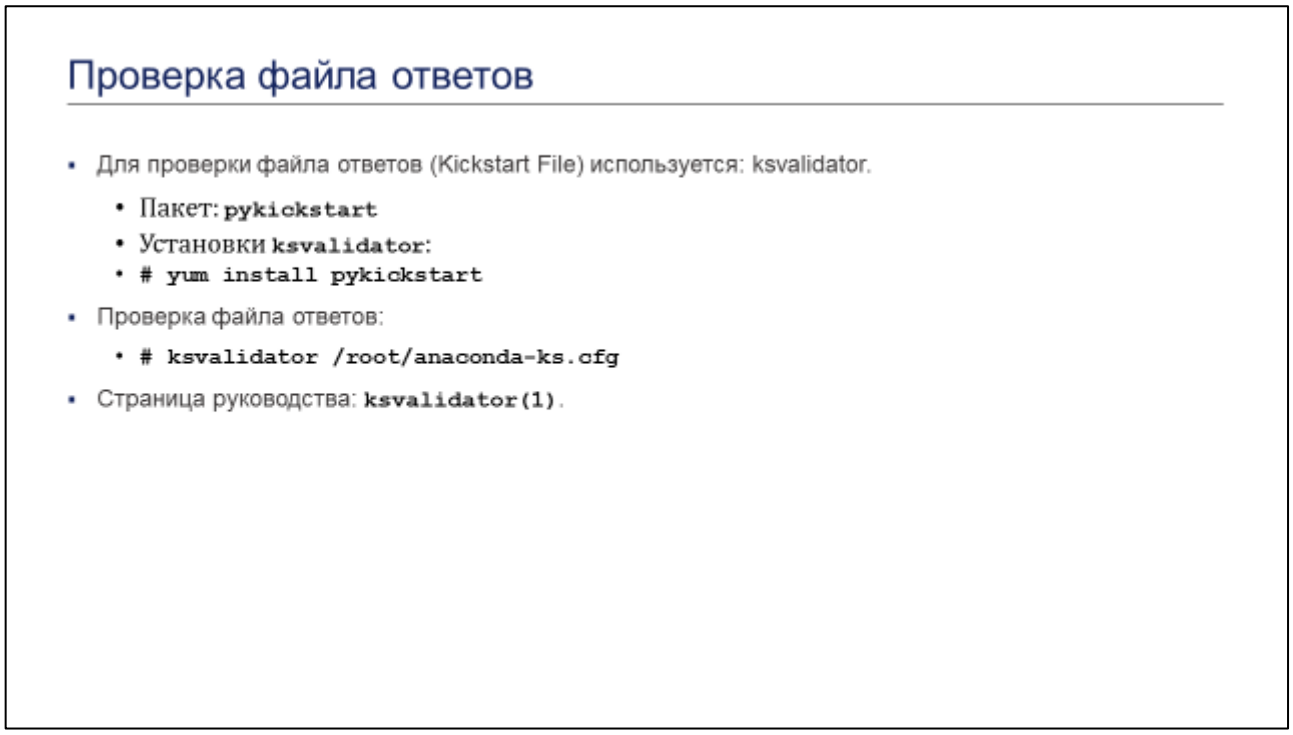
Для установки Kickstart Configurator, воспользуйтесь командой:

```
# yum install system-config-kickstart -y
```



При создании файла ответов (Kickstart File), нужно помнить следующее:

- Секции должны быть указаны в порядке.
- Элементы секции не имеют порядка.
- Не обязательные элементы могут быть пропущены.
- Пропуск обязательного элемента заставит программу установки запросить ответ на нужный вопрос.
- Строки, которые начинаются со знака решетки (**#**) – воспринимаются как комментарии и игнорируются.



После создания и настройки фала ответов (Kickstart File), полезно проверить файл перед началом выполнения установок с его использованием. RedOS 7 содержат утилиту командной строки **ksvalidator**, которая позволяет проверять файлы ответов. Для установки **ksvalidator**, воспользуйтесь следующей командой:

```
# yum install pykickstart
```

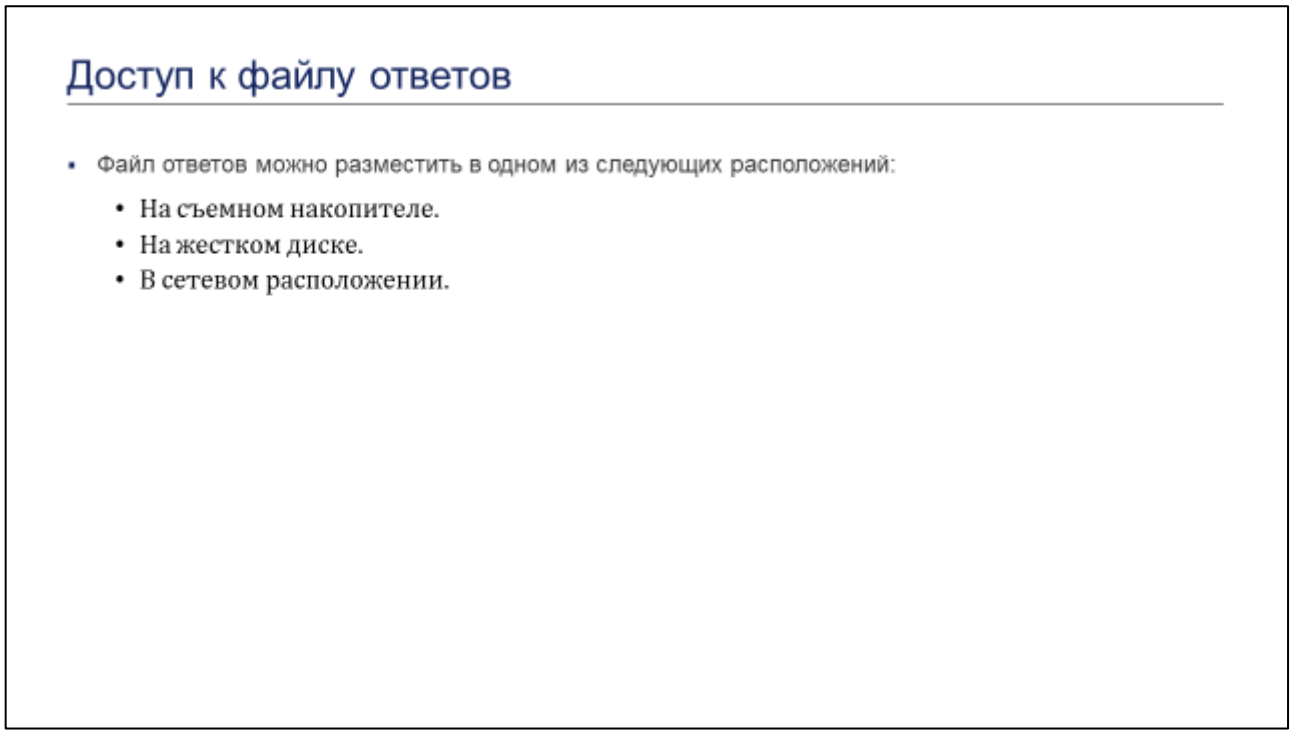
Для проверки файла ответов (Kickstart File) воспользуйтесь следующей командой:

```
# ksvalidator /root/anaconda-ks.cfg
The following problem occurred on line 4 of the kickstart file:

Unknown command: REDOS-102
```

Для получения дополнительной информации об этом инструменте обратитесь к странице руководства **ksvalidator(1)**.

ПРИМЕЧАНИЕ

Утилита **ksvalidator** имеет ряд ограничений. Файл ответов (Kickstart File) может быть очень сложным, **ksvalidator** может проверить корректность синтаксиса и отсутствие устаревших опций, но она не может гарантировать успешную установку с использованием этого файла. Также **ksvalidator** не может проверить секции **%pre**, **%post** и **%packsages** в файле ответов (Kickstart File).



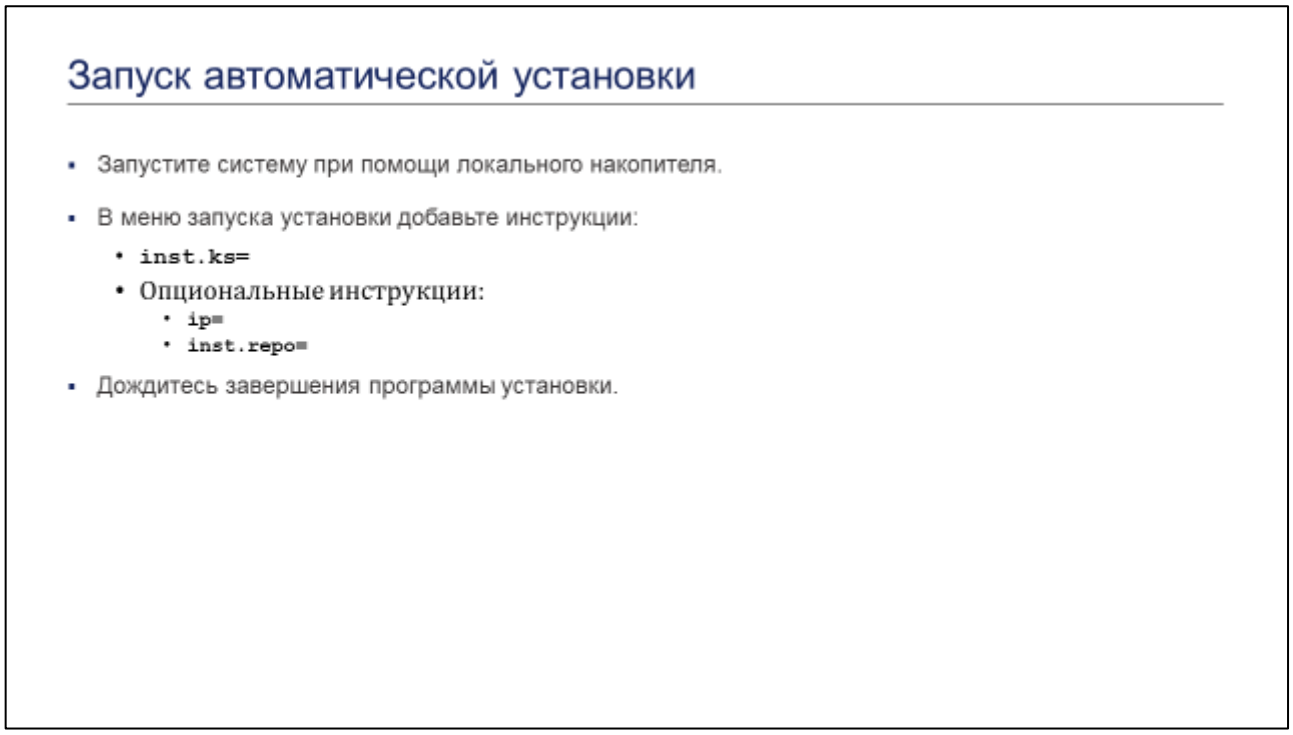
Файл ответов (Kickstart File) можно разместить в одном из следующих расположений:

- На съемном накопителе, таком как DVD-диск или флэш устройство USB.
- На жестком диске, подключенном к устанавливаемой системе.
- В доступном для устанавливаемой системы сетевом расположении.

Наиболее часто файл ответов размещают в сетевом расположении, что позволяет его использовать в комплексной установке по сети с использованием PXE-загрузки.

ПРИМЕЧАНИЕ

Для автоматической загрузки файла ответов (Kickstart File) без указания опции загрузки **`inst.ks=`**, файл с именем **`ks.cfg`** должен располагаться на тому с меткой **`OEMDRV`**.



Для запуска автоматической установки, необходимо указать опцию загрузки (**inst.ks=**) при запуске программы установки.

Системы AMD64 и Intel 64 обладают возможность загрузки по сети с использованием PXE-сервера. При настройке PXE-сервера, опции загрузки можно указать в конфигурационном файле, которые позволяет выполнить установку автоматически. Используя этот подход, можно полностью автоматизировать процесс установки, в том числе процесс загрузки.

Запуск процедуры автоматической установки вручную (Kickstart Installation).

Процедура запуска автоматической установки (Kickstart Installation) при помощи опций загрузки (указанных в строке **boot:**), состоит из следующих шагов:

1. Запустите систему при помощи локального накопителя (CD, DVD или USB-накопитель).
2. В строке загрузки укажите опцию загрузки **inst.ks=** и расположение файла ответов (Kickstart File). Если файл ответов (Kickstart File) находится в сетевом расположении, необходимо сконфигурировать сети при помощи опции **ip=**. В некоторых случаях также необходимо указать расположение источника установочных файлов, при помощи опции **inst.repo=**.
3. Запустите процесс установки вместе с добавленными опциями загрузки.