ЛИЧИ ТЕХНОЛОГИИ

КУРС ОБУЧЕНИЯ СИСТЕМНОГО АДМИНИСТРАТОРА

дистрибутива

OpenScaler Linux

Справочная книга

Версия 1.0

OpenScaler

Этот курс был разработан экспертами компании ЛИЧИ Технологии. Возможны орфографические ошибки и опечатки, как это часто бывает в больших литературных работах. Если вы заметите неточность при чтении, мы приносим свои извинения и будем благодарны, если вы сообщите нам по адресу web@lichi-tech.ru.
Спасибо!

1. ВВЕДЕНИЕ

1.1 О ДАННОЙ КНИГЕ

Данная книга представляет собой первое издание комплекта учебных материалов, разработанных компанией ООО “Личи Технологии” в содружестве с техническим комитетом сообщества разработки свободного дистрибутива OpenScaler Linux. Она ориентирована в первую очередь на студентов технических вузов и специалистов, желающих получить базовые знания работы с операционными системами Linux и подготовиться к прохождению экзаменов на сертификацию “системного администратора OpenScaler Linux”. Учебное пособие также сопровождается методическими указаниями с перечнем практических лабораторных работ, исполняемых учащимся самостоятельно с целью закрепления материала теоретического курса, представленного в данной книге. Базовый курс ориентирован на освоение базовых вопросов установки, первичной настройки и администрирования операционной системы в условиях корпоративной среды. В частности в составе курса рассмотрены вопросы установки ОС, управления пользовательскими ролями и доступом, управление дисковой подсистемой, сетевыми устройствами, осуществление конфигурирования различных компонентов и сервисов операционной системы в консольном режиме работы с использованием текстовых редакторов и путем редактирования конфигурационных файлов.

Обучающий курс, представленный в данной книге, также будет полезен специалистам Linux систем, ранее работавшим с альтернативными дистрибутивами ОС и желающими расширить свои знания.

Все практические задания проводятся на последний на момент выпуска данной книги версии стабильной (LTS) версии дистрибутива - OpenScaler 22.03 LTS SP2, поддерживающей архитектуры X86_64 и ARM, доступной для свободной загрузки с сайта открытого сообщества разработчиков дистрибутива по ссылке https://openscaler.ru/downloads/

1.2 О OPENSCALER LINUX

В последнее десятилетие российские организации, как государственные и научно-образовательные учреждения, так и коммерческие компании, столкнулись с большим количеством проблем в связи с уходом с российского рынка многих западных разработчиков аппаратного и программного обеспечения. Программные и аппаратные решения зачастую подлежат вынужденному “импортозамещению” на несанкционные альтернативные варианты, зачастую уступающие по функциональности и производительности.

Одним из главных ударов стал уход с российского рынка компании Red Hat — мирового лидера в сфере разработки свободного программного обеспечения для организаций. Ведь не секрет, что большинство крупных российских организаций вне зависимости от сферы их деятельности давно и активно используют операционную систему Red Hat Enterprise Linux (RHEL), платформу оркестровки контейнеризированных приложений

Red Hat OpenShift, связующее программное обеспечение линейки JBoss и другие инфраструктурные и платформенные продукты вендора. Оказавшись без возможности получения обновлений программного обеспечения и услуг технической поддержки, многие организации находятся в поиске альтернатив, способных (в идеале — полностью) заменить продукты американской компании, да так, чтобы эта вынужденная замена была проведена с минимальными накладными расходами, с минимальным простоем критически важных корпоративных сервисов и без необходимости переподготовки своих IT-специалистов для работы с новым решением.

Стоит отметить, что и единственный наиболее близкий некоммерческий аналог Red Hat Enterprise Linux — дистрибутив CentOS — также фактически прекратил существование в исходном виде, утратив совместимость с коммерческой версией и став, по сути, полигоном для апробации новых технологий Red Hat (CentOS Stream). С прекращением развития CentOS задача поиска альтернативной операционной системы стала насущной и для российского малого и среднего бизнеса, зачастую использующего данную систему; тем более, что недавно появившиеся дистрибутивы Alma и Rocky, призванные заменить CentOS, пока ещё не проверены временем и перспективы их развития туманны.

Для решения данной проблемы силами нескольких специалистов, имеющих большой опыт работы со свободным программным обеспечением и продуктами Red Hat в частности, было создано независимое сообщество OpenScaler. Задача нашего сообщества состоит в адаптации дистрибутива openEuler к требованиям крупных российских организаций с целью предоставления программного решения максимально совместимого с Red Hat Enterprise Linux, а также имеющего ряд функциональных и технических преимуществ перед ним. Сообщество основано IT-специалистами, имеющими как минимум десятилетний опыт работы с передовым свободным программным обеспечением в роли как инженеров-программистов, архитекторов IT-инфраструктур, так и специалистов по техническому сопровождению и эксплуатации сложных программно-аппаратных комплексов, на основе СПО. Ключевым продуктом сообщества является свободный некоммерческий дистрибутив openScaler OS, являющийся локализованной версией дистрибутива openEuler.

Но почему же новое зарождающееся сообщество в качестве ключевой альтернативы выбрало, казалось бы, мало кому известный дистрибутив, спросите вы? Учитывая проблематику и потребности российских компаний, был проведен всесторонний анализ существующих на текущий момент ключевых некоммерческих дистрибутивов GNU/Linux с целью определения максимально соответствующего кандидата на роль «заместителя» RHEL. Отбор проводился на основе множества критериев включавших, в частности, такие, как наличие успешных коммерческих ответвлений, количество участников активного сообщества разработки, уровень документального сопровождения, функционально-технические особенности реализации, состав пакетной базы и многих других.

На сегодняшний день openEuler это:

- единая система для всех архитектур оборудования (ARM, RISC-V, x86-64) позволяет унифицировать инфраструктуру ЦОДа, сведя ее к одной версии операционной системы, и обеспечить безболезненный перенос инфраструктуры с одной аппаратной платформы на другую;
- предназначена для решения задач корпоративного уровня, в том числе для сценариев с базами данных, большими данными, облачными вычислениями, системами искусственного интеллекта;
- ключевая операционная система для программ импортозамещения в Китае. Решения на базе openEuler OS активно внедряются и используются в критических инфраструктурах банков, телекоммуникационных компаний, организаций сектора госуправления. В создании экосистемы openEuler участвуют более 16 тысяч разработчиков;
- включает передовые программные наработки:
    - **A-Tune** – систему для автоматической оптимизации настроек с помощью механизма машинного обучения. С помощью технологий искусственного интеллекта подбираются оптимальные параметры конфигурации операционной системы для повышения общей эффективности работы системы в соответствии с рабочей нагрузкой;
    - собственный стек легковесной виртуализации (**StratoVirt**) и контейнеризации (**iSulad**);
    - инструментарий разработки доверенных приложений **secGear**;
    - инструмент **secPaver** для настройки политик информационной безопасности **SELinux**;
    - множественные оптимизации для архитектуры ARM, в частности, серверов Huawei Taishan;
    - **NFS multipathing** разработана для устранения предыдущих дефектов, возникающих при использовании традиционной NFS;
    - Технология **Kernel Live Upgrade**, позволяющая обновить ядро операционной системы и произвести перезапуск системы Linux без фактического перезапуска сервера;
    - Технология **SysCare**, которая поможет произвести установку патча для процесса пользователя без его перезапуска;
    - собственное решение **x2openEuler** для оценки сложности миграции со сторонних операционных систем, способное рассматривать разные сценарии (новое развертывание с нуля, расширение существующего решения, замена на месте), а также определяет точки изменения для миграции и повышает эффективность перехода с точки зрения совместимости программного обеспечения, оборудования и элементов конфигурации. Таким образом, x2openEuler позволяет оценить весь путь миграции: общий анализ дизайн решения портирование и адаптация реализация замены тестирование и запуск.

1.3 ИСТОРИЯ OPENEULER

openEuler – это свободный дистрибутив операционной системы GNU/Linux с открытым исходным кодом. Основан на собственном дистрибутиве Huawei Euler Linux OS.

Изначально Euler OS была запущена Huawei в 2010 году и до 2015 года имела статус ОС для внутреннего корпоративного использования. В 2016-17 гг. была выпущена версия для внешнего коммерческого использования. И, наконец, в июле 2019 г. на ее основе была выпущена open-source версия – openEuler, разработанная и на сегодняшний день управляемая китайской open-source организацией «OpenAtom Foundation».

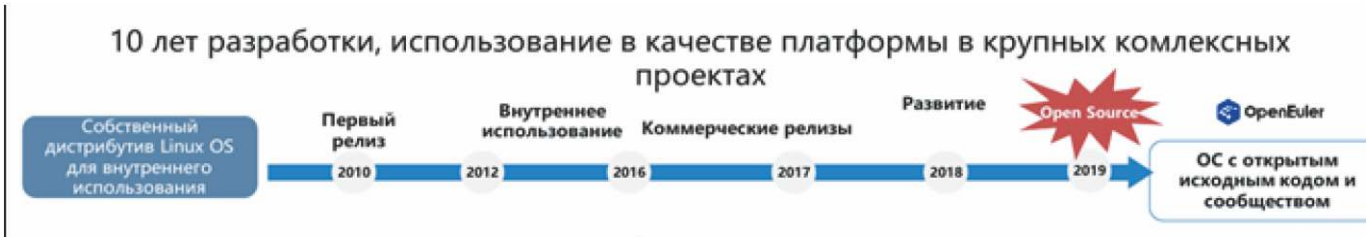

Рисунок 1. Вехи развития дистрибутива openEuler

Генеалогия дистрибутива: Red Hat Enterprise Linux -> CentOS -> Euler Enterprise Linux OS -> openEuler Linux OS (openEuler).

На текущий момент openEuler развивается независимым открытым сообществом разработчиков (https://ru.openeuler.org/ru/) и рассматривается как единая программная платформа под все ключевые задачи, сферы применения и архитектуры оборудования (поддержка ARM, RISC-V, X86, Ascend).

openEuler является кросс-платформенной системой, уделяет особое внимание поддержке платформ с процессорами на базе архитектур ARM и RISC–V и предназначена для решения enterprise-задач, в том числе для сценариев с базами данных (Data Base), большими данными (Big Data), облачными вычислениями (Cloud Computing) и ИИ (AI).

Рисунок 2. openEuler: единая ОС для всех архитектур и ключевых корпоративных сценариев использования

Ключевые преимущества:

1. openEuler OS обеспечивает максимальную совместимость с RedHat/CentOS и предоставляет готовые инструменты для миграции с других ОС.

В конце 2021 года RedHat Enterprise Linux поменял концепцию CentOS, в результате чего корпоративные заказчики остались без свободного дистрибутива, не требующего покупки платной подписки на техническую поддержку. При этом переход на платную техническую поддержку невозможен, поскольку с марта 2022 года RedHat Enterprise Linux в России более не работает.

2. openEuler OS – поддержка на межгосударственном Российско-Китайском уровне.

Договоренность о совместной работе над взаимным развитием open-source экостистем и в частности ОС openEuler закреплена в двустороннем коммюнике премьер-министров России и Китая.

3. openEuler OS – поддерживает различные архитектуры процессоров

Поддержка различных аппаратных архитектур (ARM, X86, RISC–V) позволяет унифицировать инфраструктуру ЦОДа, сведя ее к одной версии ОС, и обеспечить безболезненный перенос инфраструктуры с одной аппаратной платформы на другую.

4. openEuler OS – ключевая ОС для программ импортозамещения в Китае.

Решения на базе openEuler OS активно внедряются и используются в критических инфраструктурах Банков, Телеком, Интернет и других компаний. Экосистема openEuler включает в себя более 500 000+ разработчиков.

5. openEuler OS – включает в себя передовые программные наработки Huawei:

- A–Tune – система для автоматической оптимизации настроек с помощью механизма машинного обучения. С помощью ИИ подбираются оптимальные параметры конфигурации операционной системы для повышения общей эффективности работы системы в соответствии с рабочей нагрузкой;
- Собственный стек решений классической (StratoVirt) и контейнерной (iSulad) виртуализации совместимой с классическими Qemu и Docker;
- Инструментарий разработки доверенных приложений secGear;
- Фирменный инструмент настройки политик информационной безопасности SELinux, AppArmor под названием secPaver;
- Множественные оптимизации для ARM серверов, в частности Huawei Taishan.

2.УСТАНОВКА И ПЕРВИЧНАЯ НАСТРОЙКА ДИСТРИБУТИВА OPENSCALER

В рамках данной главы мы научимся проводить штатную установку дистрибутива OpenScaler. На данный момент пользователям на выбор предоставляется две версии дистрибутива:

- инновационная версия, выходящая каждые полгода и включающая в себя наиболее новые версии ПО, ядра ОС и зачастую новые функции отсутствующие в стабильной версии дистрибутива. Данная версия является экспериментальной и используется членами сообщества разработчиков и энтузиастами в основном для апробации новых версий пакетов и функционала. Она не рекомендована для установки в продуктивную среду и использования для предоставления Mission-Critical сервисов в корпоративной инфраструктуре.
- стабильная версия, имеющая префикс LTS (Long Time Support). Выходит раз в 3 года и включает только прошедшие проверку на стабильность и надежность версии ядра и программного обеспечения, тем самым предоставляя требуемый уровень надежности для корпоративного использования.

С точки зрения процесса установки - различия в версиях ни коим образом не сказываются на последовательности действий пользователя. Таким образом, для прохождения данного курса, обучающийся волен сам выбирать версию дистрибутива. Далее все действия в данной книге будут описаны на примере последней на момент написания стабильной версии - 22.03 LTS SP2.

2.1 МИНИМАЛЬНЫЕ ТРЕБОВАНИЯ К ОБОРУДОВАНИЮ/ВИРТУАЛЬНОЙ МАШИНЕ

OpenScaler поддерживает архитектуры X86_64 и ARM. Поскольку процессоры данных архитектур различаются по набору команд, обучающийся должен самостоятельно загрузить с сайта сообщества дистрибутив (установочный ISO образ) подходящий для того типа оборудования на которое планируется осуществить установку.

Установочные образы доступны на сайте сообщества разработчиков по ссылке https://openscaler.ru/downloads/

С целью выполнения учебных практических работ, в рамках данного курса обучающемуся предлагается использовать один из следующих вариантов платформ для установки дистрибутива

- Стандартный персональный компьютер архитектуры X86_64 с установленным программным обеспечением виртуализации, например VirtualBox, VmWare Workstation и пр.
- Физический сервер архитектуры ARM или X86_64. В случае выбора данного подхода обучающийся должен иметь достаточные навыки для работы в IPMI интерфейсе выбранного сервера и уметь производить настройку данного оборудования самостоятельно. Вопросы настройки серверного оборудования не покрываются данным курсом, ориентированным исключительно на вопросы установки и настройки самой операционной системы.

Вне зависимости от выбранного типа аппаратного обеспечения для установки операционной системы необходимо обеспечить следующие минимальные требования для корректности работы.

Таблица 1. Минимальные требования для установки ОС

Компонент	Минимальные требования	Описание
Архитектура	AARCH64 X86_64	64-х битные архитектуры ARM и X86
Процессор	Двухядерный процессор	При работе в виртуализованной среде на базе Virtual Box или Vmware Workstation рекомендуется использовать минимум 4х ядерные процессоры
ОЗУ	не менее 4 ГБ	При работе в виртуализованной среде на базе Virtual Box или Vmware Workstation рекомендуется иметь 16ГБ ОЗУ для корректной работы гипервизора
Жесткий диск	не менее 32 ГБ	

2.2 ПРОВЕРКА ЦЕЛОСТНОСТИ ЗАГРУЖЕННОГО УСТАНОВОЧНОГО ISO ОБРАЗА

С целью предотвращения некорректных установок дистрибутива ввиду неполной или некорректной загрузки установочного образа из-за проблем с сетью или запоминающим устройством во время передачи данных, следует выполнить следующие шаги для проверки целостности перед фактическим осуществлением установки:

1. Получите проверочное значение в файле проверки выполнив следующую команду:
   $cat OpenScaler-22.0.3-x86_64-dvd.iso.sha256sum
2. Рассчитайте проверочное значение SHA256 для загруженного файла выполнив следующую команду:
   $sha256sum OpenScaler-22.0.3-x86_64-dvd.iso.sha256sum
3. После выполнения команды отображается проверочное значение. Проверьте, совпадают ли значения, рассчитанные на шаге 1 и шаге 2.

Если проверочные значения совпадают, файл установочный образ ISO не поврежден и его можно использовать для проведения установки операционной системы. Если же они не совпадают, файл поврежден, и обучающемуся необходимо провести его повторную загрузку согласно ранее описанным шагам в данной книге.

2.3 ПРОВЕДЕНИЕ ШТАТНОЙ УСТАНОВКИ ДИСТРИБУТИВА OPENSCALER

Для установки дистрибутива возможно использование как USB-Flash дисков так и физических DVD дисков или монтирование установочного ISO образа в виртуальном DVD-Rom (к примеру, в виртуальной машине). Осуществив загрузку, на экране будет отображаться меню загрузчика дистрибутива как представлено на рисунке 3.

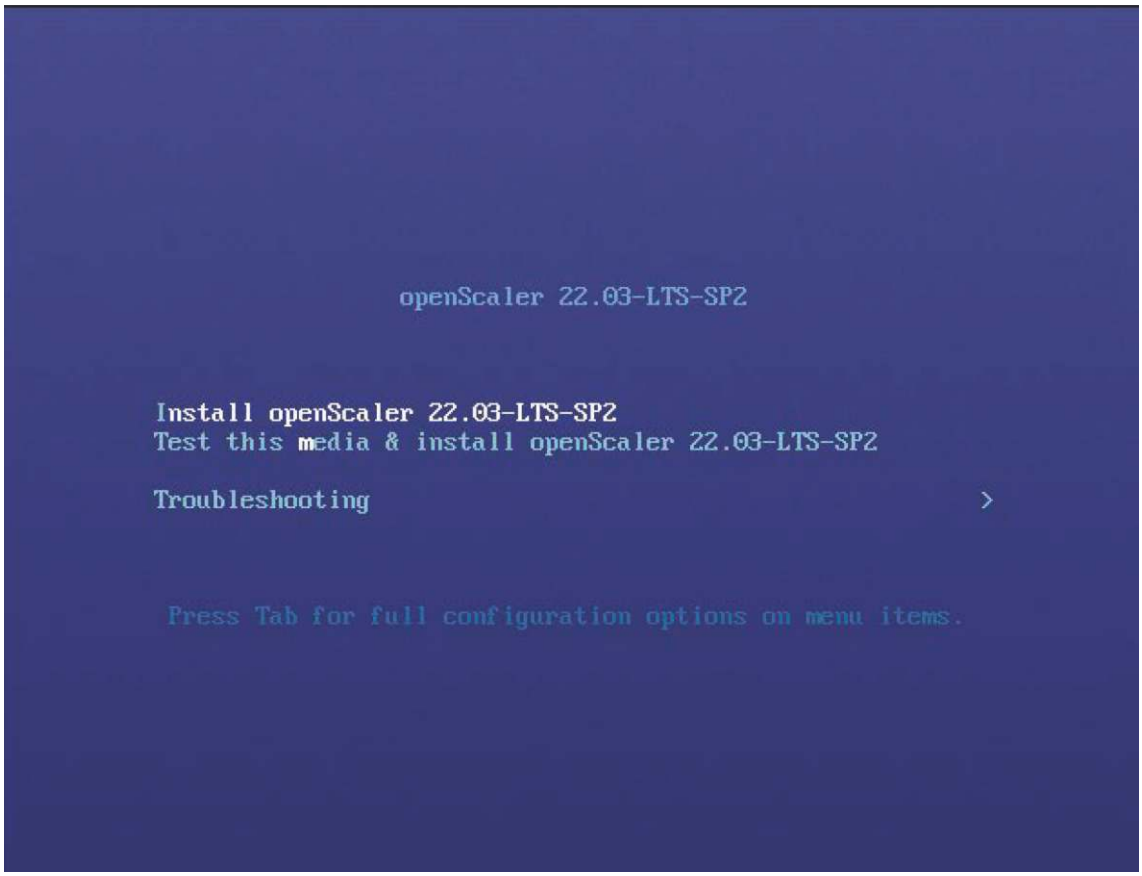

Рисунок 3. Экран первой загрузки с установочного образа OpenScaler

По умолчанию будет выделена опция установки дистрибутива OpenScaler. По желанию, вы можете выбрать вторую строку в загрузчике (выбор строк осуществляется стрелками “вверх/вниз”) которая перед установкой дистрибутива предварительно проверить корректность носителя данных во избежание проблем с установкой. Выберите требуемые вариант и нажмите клавишу “ввод”.

После перехода на страницу установки с графическим пользовательским интерфейсом выполните следующие операции для установки дистрибутива:

- Укажите язык установки. По умолчанию используется русский язык. Вы можете изменить язык в соответствии с конкретными предъявляемыми требованиями, как показано на рисунке 4. По завершении выбора языка нажмите на кнопку “продолжить”.

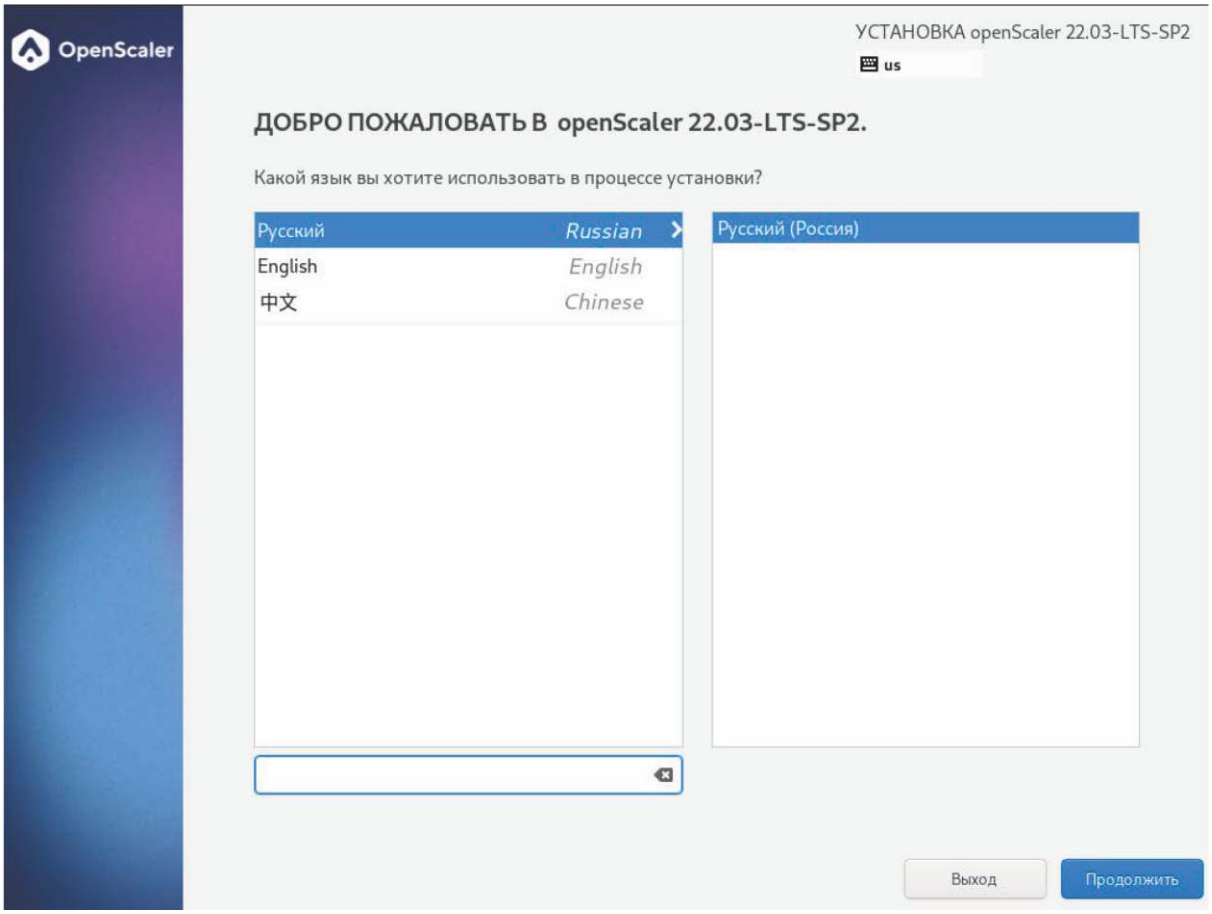

Рисунок 4. Выбор языка установки дистрибутива

- На странице “Обзор установки” задайте элементы конфигурации в соответствии с конкретными предъявляемыми требованиями:

  - Элемент конфигурации с символом предупреждения необходимо настроить в обязательном порядке. Без настройки данных компонентов установка программного продукта не может быть продолжена в штатном режиме, и, соответственно кнопка “Начать установку” будет неактивной (отображается на белом фоне). После настройки компонента символ предупреждения исчезнет, вы можете выполнить следующую операцию.
  - Элемент конфигурации без символа предупреждения настроен по умолчанию и не требует своей обязательной настройки и/или изменения значений на момент установки продукта..
  - Вы можете нажать кнопку “Начать установку” для проведения штатной установки системы, только когда все сигналы предупреждения будут устранены.

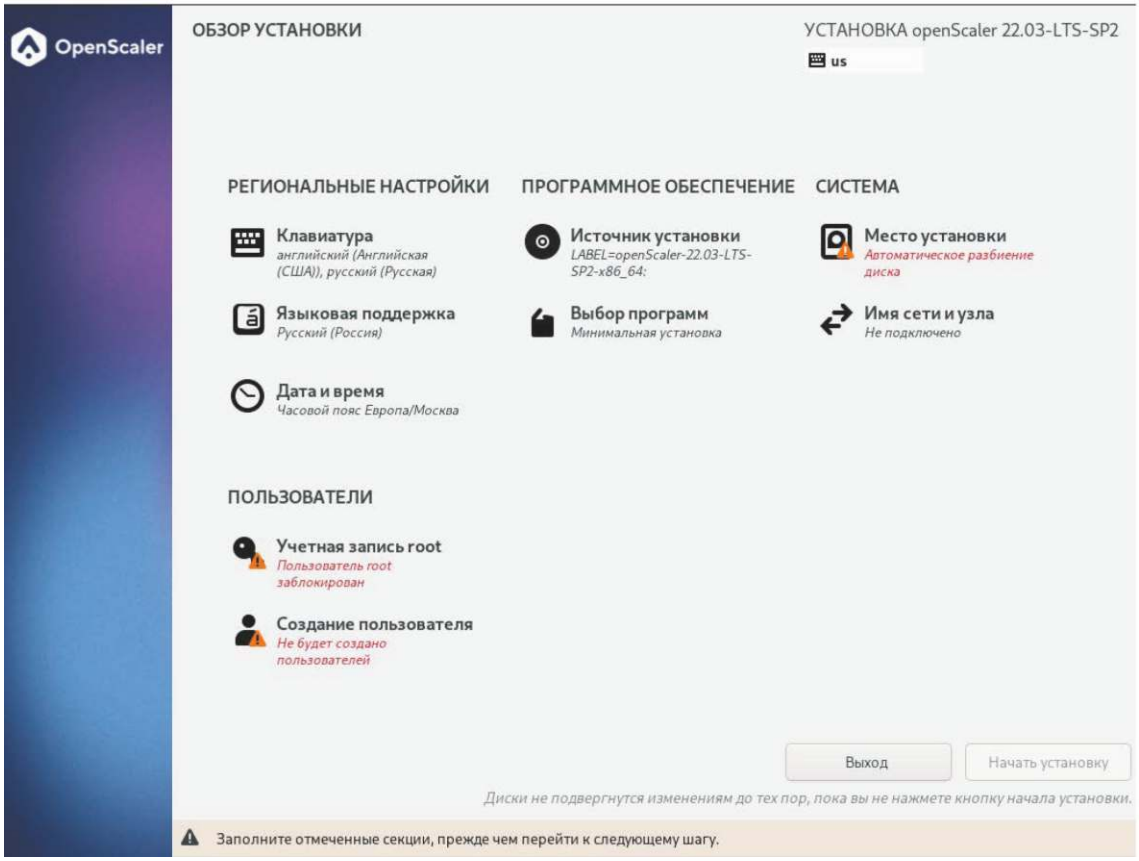

Рисунок 5. Сводка по параметрам установки

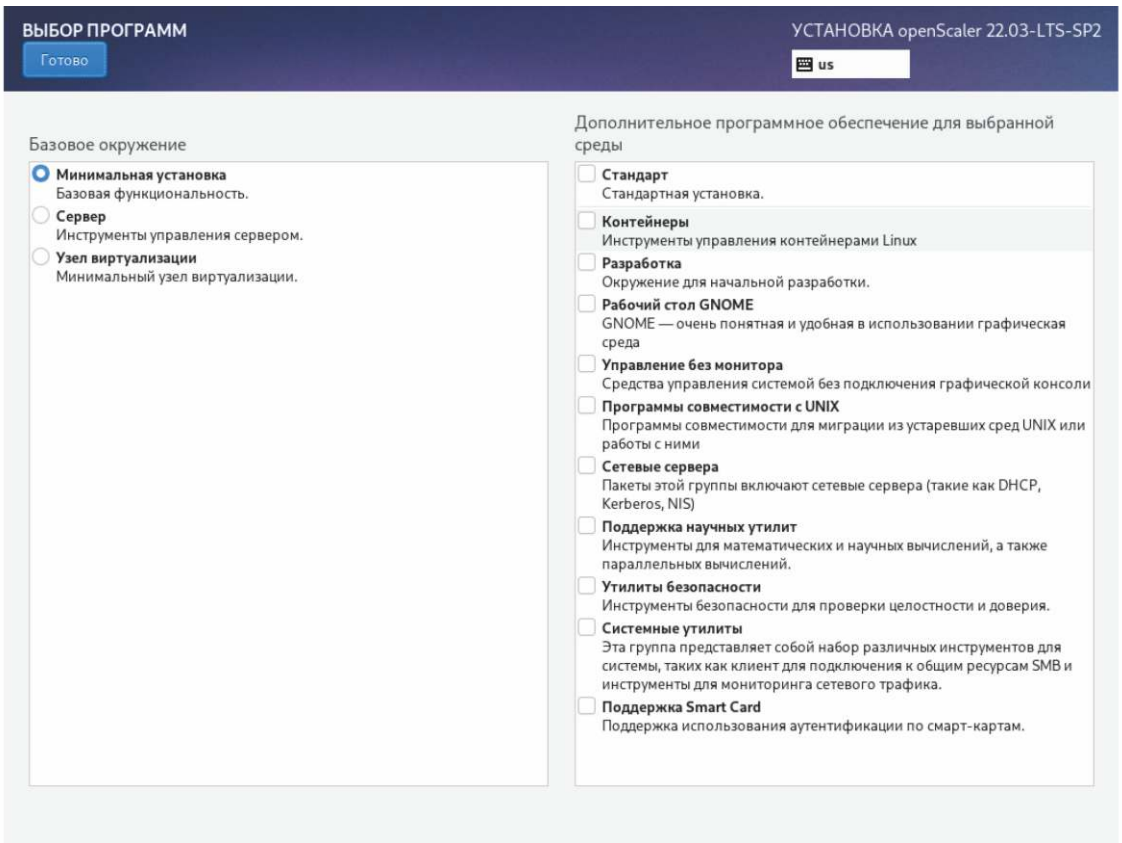

Рисунок 6. Выбор базового окружения и состава программных пакетов для установки

- Выберите элемент “Выбор программ”, чтобы задать элементы конфигурации.
- Учитывая конкретные предъявляемые требования, выберите требуемый сценарий использования дистрибутива (в рамках курса будет выбран вариант “Минимальная установка”) в левой области именуемой “Базовое окружение” и требуемый набор программных пакетов для установки в правой области, как показано на рисунке 6.

ПРИМЕЧАНИЕ

- В режиме **Минимальной установки** устанавливаются не все пакеты доступные на установочном образе дистрибутива. Если по завершению установки продукта потребуется доустановить дополнительный программный пакет, вы можете подключить установочный образ в качестве репозитория программного обеспечения и провести штатную доустановку необходимых программных компонентов используя встроенный в продукт менеджер программных пакетов именуемый DNF .
- Если выбрать режим установки **Узел виртуализации**, в дополнение к базовым компонентам продукта будут доустановлены программные решения обеспечивающие функционал виртуализации, к примеру, такие компоненты как QEMU, libvirt и edk2 будут установлены по умолчанию.

- Завершив настройку, нажмите кнопку “Готово” в левом верхнем углу, чтобы вернуться на страницу “Обзор установки”.
- Выберите “Место установки”, чтобы задать элементы конфигурации.
- На странице “Место установки” выберите локальное запоминающее устройство.

ПРИМЕЧАНИЕ

- При выборе устройства для установки также нужно настроить хранилище для создания разделов в системе. Можно настроить разделы вручную или выбрать **Автоматически** для автоматического их создания.

- Выберите “Автоматически” если программное обеспечение устанавливается на новом запоминающем устройстве либо хранить данные на этом запоминающем устройстве не требуется, как показано на рисунке 7.

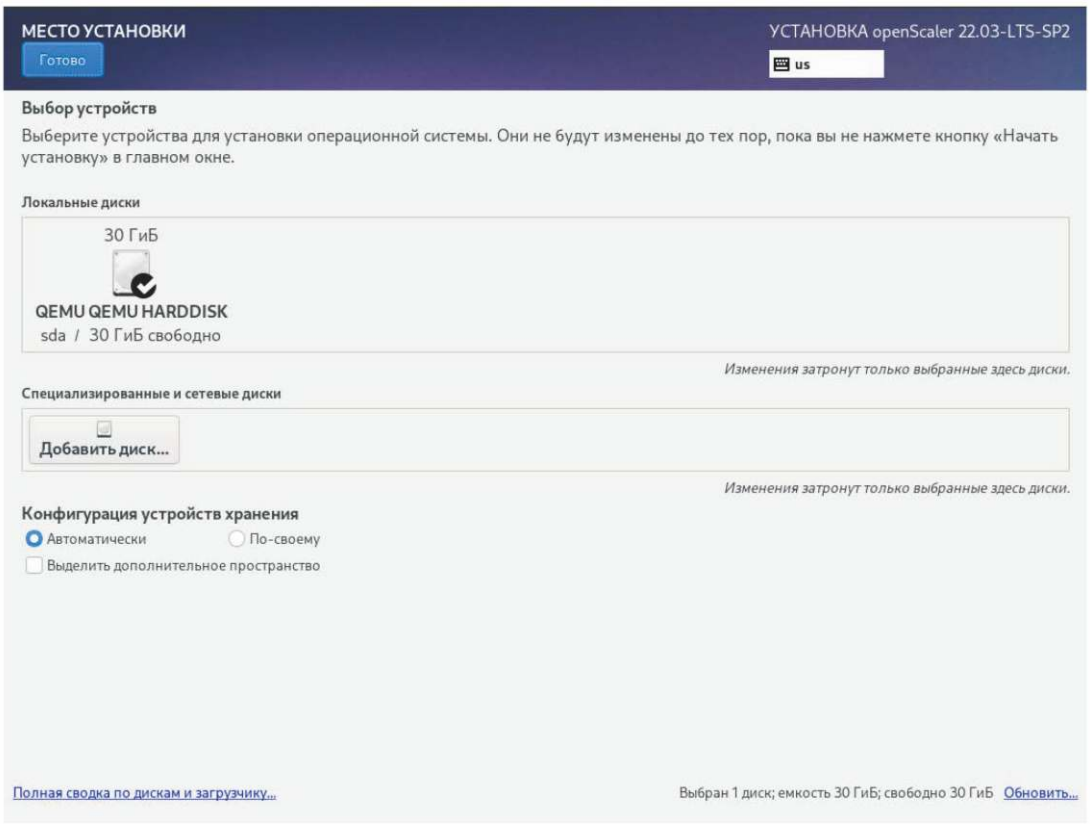

Рисунок 7. Выбор устройства для установки операционной системы

ПРИМЕЧАНИЕ

Если для системы настроен раздел подкачки, он используется при нехватке физической оперативной памяти системы. Хотя раздел подкачки можно использовать для расширения физической оперативной памяти, когда этот раздел используется из-за нехватки памяти, скорость реагирования и производительность системы снижаются. Поэтому не рекомендуется настраивать раздел подкачки в системе с достаточным объемом физической памяти или в системе, чувствительной к производительности.

Если вам нужно разделить группу логических томов, выберите “По-своему”, чтобы вручную разбить группу логических томов. На странице “Разметка вручную” нажмите на + (плюс) и добавьте необходимые разделы и точки монтирования. Завершив настройку, нажмите кнопку “Готово” в левом верхнем углу, чтобы вернуться на страницу “Обзор установки”.

- Выберите “Пароль Root” и установите пароль учетной записи root. – Данная учетная запись является основным системным администратором операционной системы никак не ограниченной в полноте своих действий.
- На странице “Пароль Root” введите пароль, соответствующий требованиям к сложности пароля, и подтвердите его, как показано на рисунке 8.

ПРИМЕЧАНИЕ

Учетная запись root используется для выполнения ключевых задач управления системой. Не рекомендуется использовать учетную запись root для повседневной работы или доступа к системе. Если выбрать “Заблокировать учетную запись root” на странице “Пароль Root”, учетная запись root будет отключена.

Пароль пользователя root или нового пользователя должен соответствовать требованиям сложности пароля. В противном случае установка пароля или создание пользователя завершатся неудачей. Пароль должен соответствовать следующим требованиям:

- Содержит не менее восьми символов.
- Содержит по меньшей мере три из следующих символов: прописные буквы, строчные буквы, цифры и специальные символы.
- Отличается от имени пользователя.
- Не содержит слова из словаря.

В OpenScaler можно выполнить команду **cracklib-unpacker /usr/share/cracklib/pw_dict > dictionary.txt,** чтобы экспортировать файл библиотеки словарей dictionary.txt. Вы можете проверить, находится ли пароль в этом словаре.

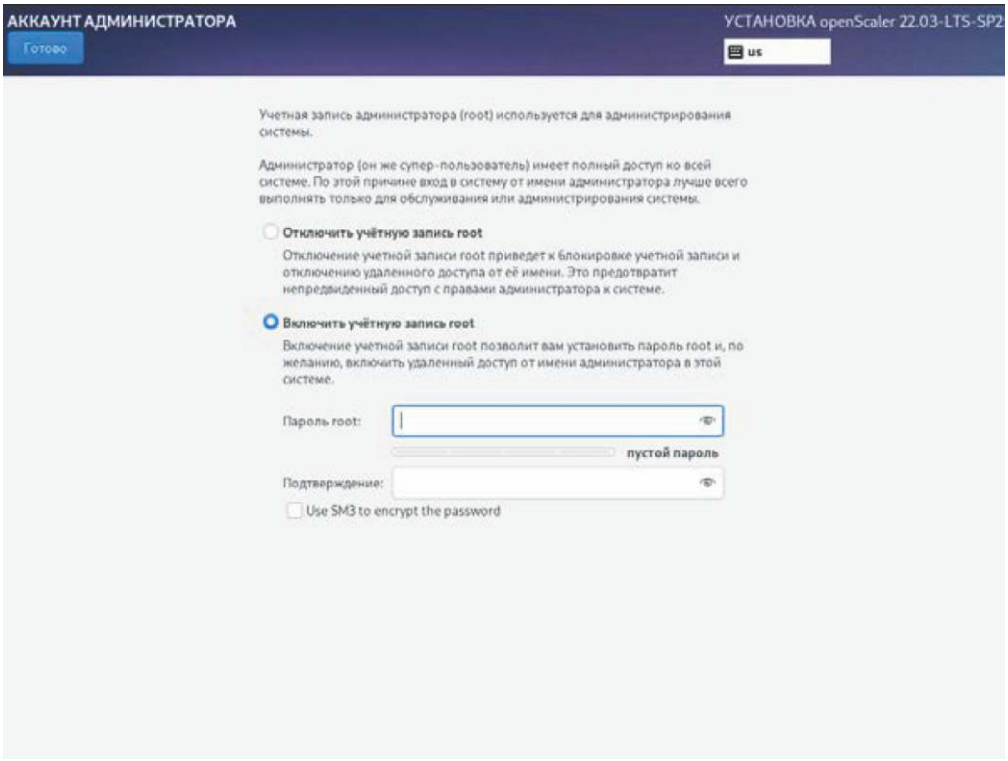

Рисунок 8. Задание пароля для учетной записи root

- После завершения настройки нажмите кнопку “Готово” в верхнем левом углу, чтобы вернуться на страницу “Обзор установки”.
- Выберите “Создание пользователя” и задайте параметры.
- На рисунке 9 показана страница создания пользователя. Введите имя пользователя и задайте пароль. Требования к сложности пароля такие же, как и для пароля учетной записи root. Кроме того, можно задать домашний каталог и группу пользователей, нажав кнопку “Дополнительно”, как показано на рисунке 10.

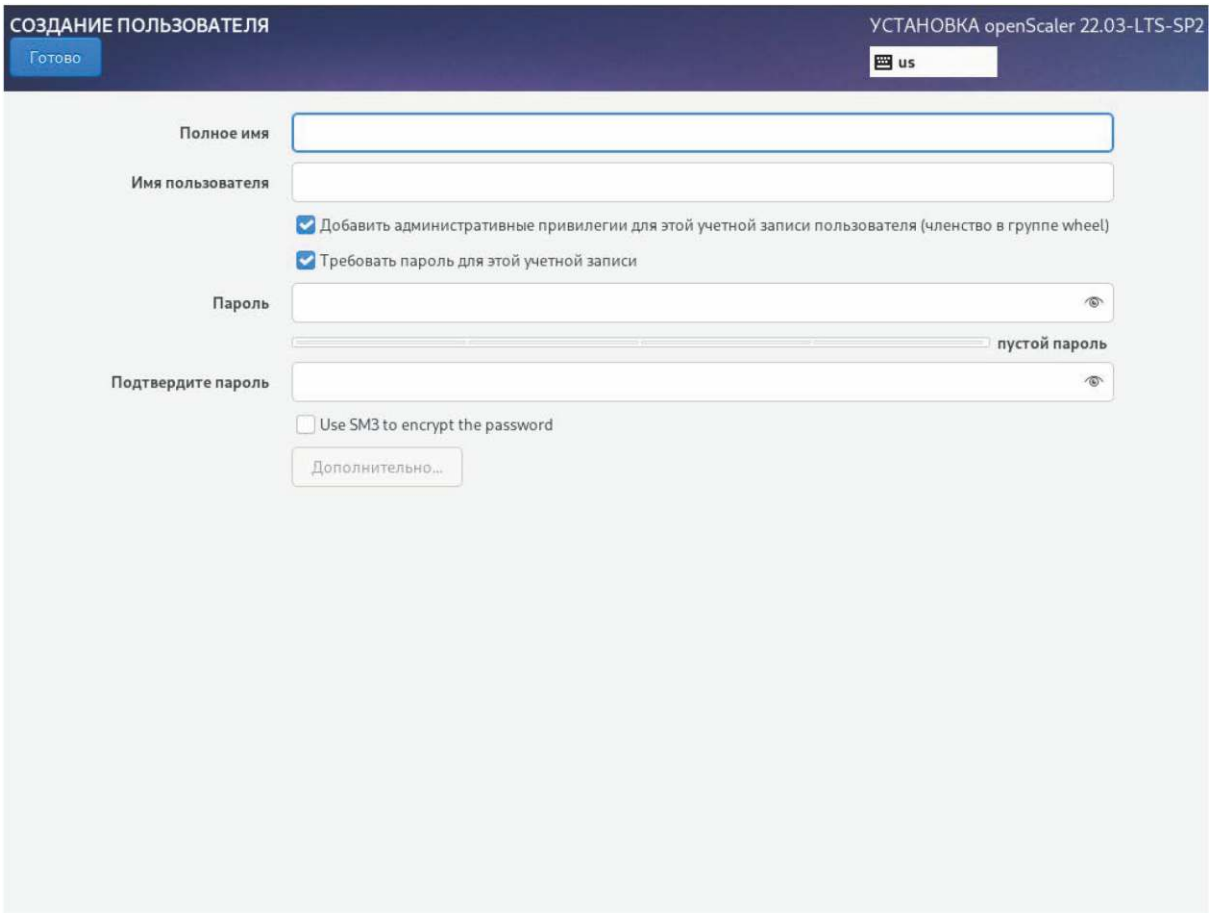

Рисунок 9. Создание учетной записи пользователя системы

- После завершения настройки нажмите кнопку “Готово” в верхнем левом углу, чтобы вернуться на страницу “Обзор установки”.
- Задайте другие элементы конфигурации. Вы можете использовать для них значения по умолчанию.
- Нажмите кнопку ‘Начать установку”, чтобы установить систему, как показано на рисунке 11.

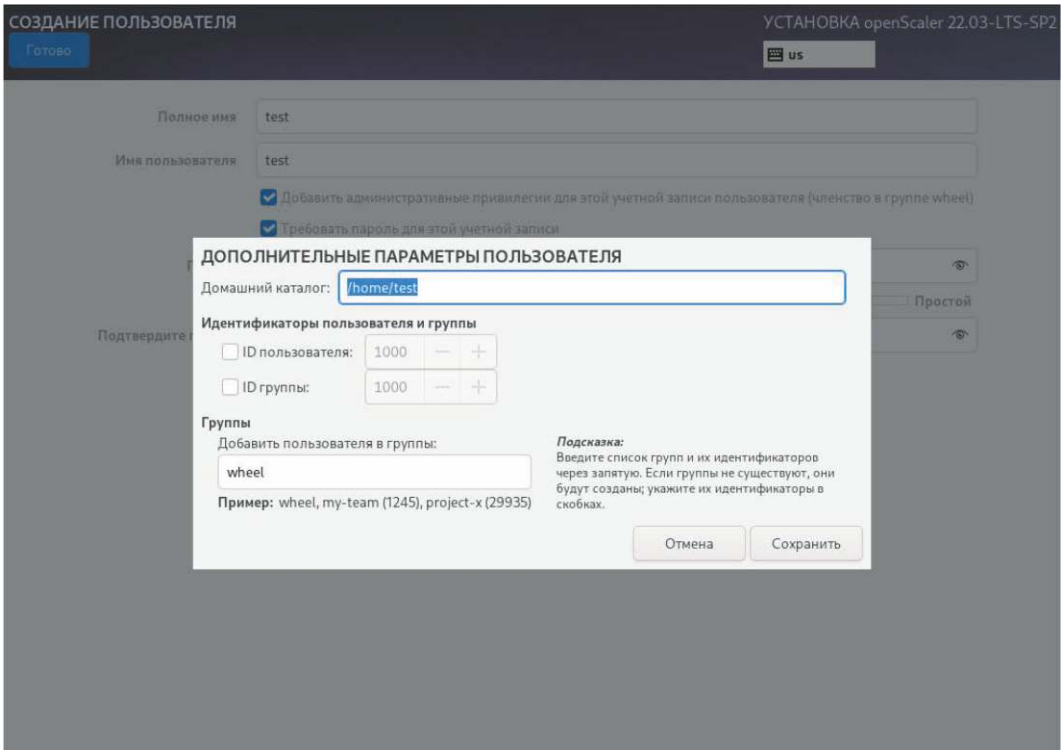

Рисунок 10. Расширенные настройки параметров пользовательской учетной записи

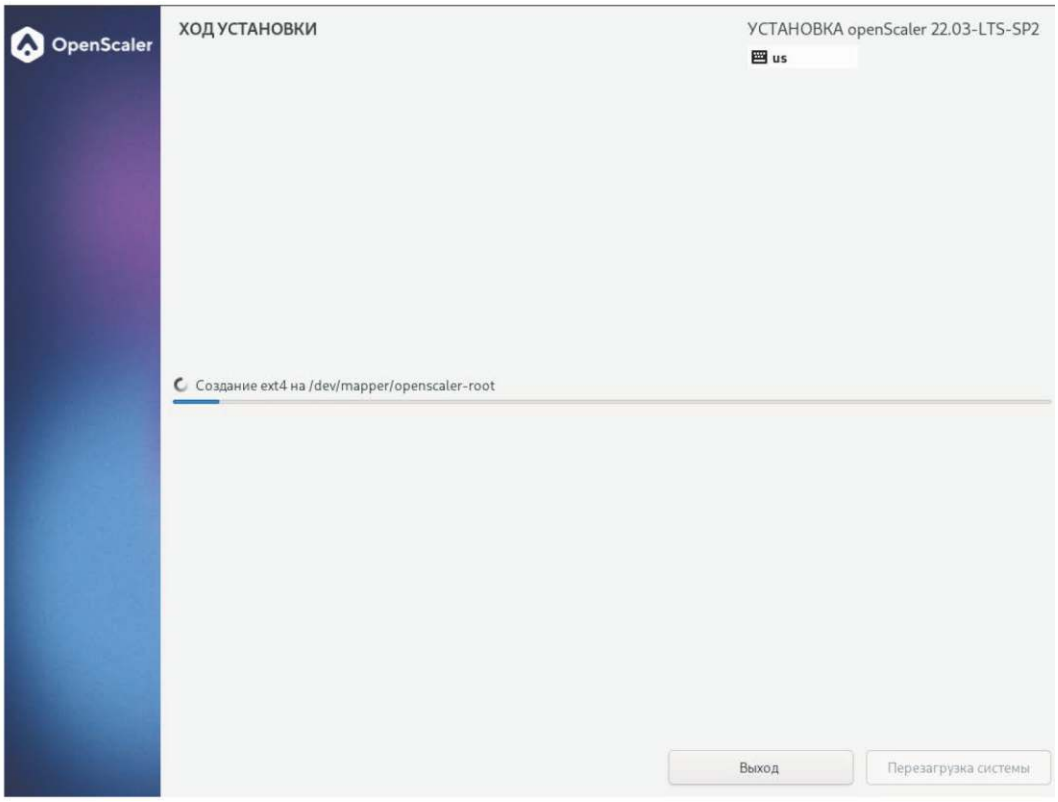

Рисунок 11. Процесс установки дистрибутива OpenScaler

- После завершения установки перезагрузите систему. Нажмите кнопку “Перезагрузить”, чтобы перезагрузить систему.

После установки и перезагрузки системы отобразится меню загрузчика операционной системы (GRUB) как представлено на рисунке 12. По умолчанию выбрана последняя версия ядра ОС, от пользователя требуется нажать клавишу “ввод” или дождаться истечения таймаута (5 секунд). После чего загрузка выбранного ядра будет осуществлена.

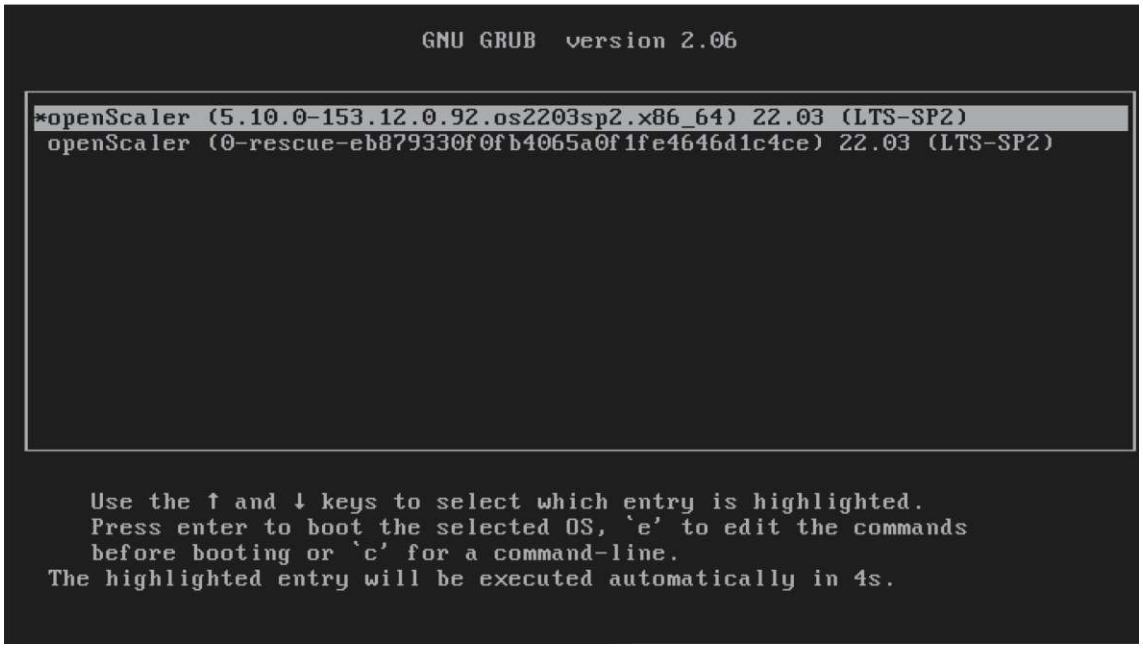

Рисунок 12. Меню загрузчика операционной системы

По завершению загрузки отображается страница входа в системный интерфейс командной строки (CLI). Введите имя пользователя и пароль, заданные во время установки, чтобы авторизоваться в системе OpenScaler. По завершению авторизации в системе, пользователь увидит базовые данные о системе и приглашение командной строки для ввода команд как представлено на рисунке 13.

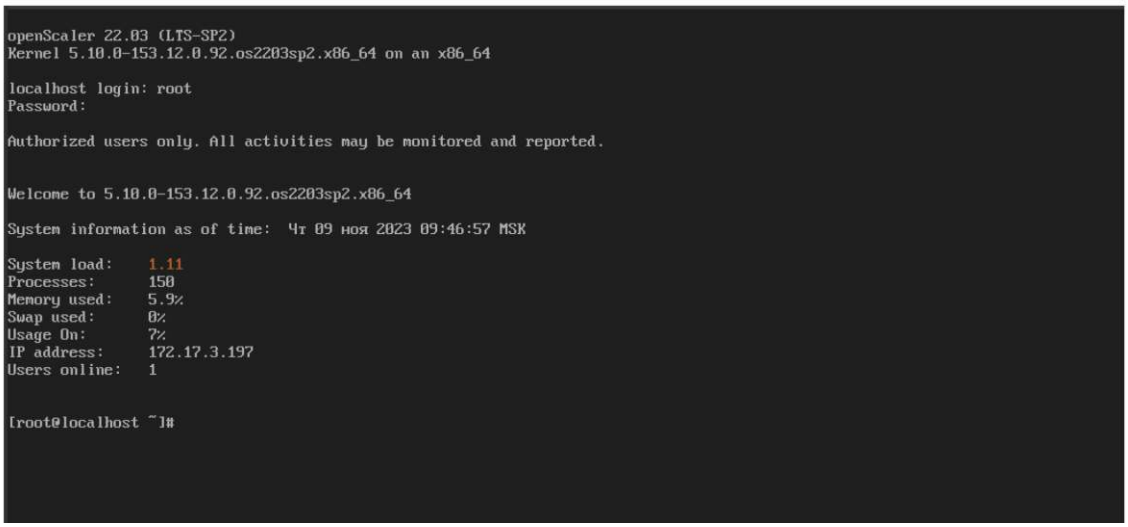

Рисунок 13. Пользователь root авторизовался в системе OpenScaler

2.4 НАЧАЛО ИСПОЛЬЗОВАНИЯ УСТАНОВЛЕННОГО ДИСТРИБУТИВА

Операционная система Linux предоставляет два варианта интерактивного окружения для взаимодействия с пользователем системы. Графический именуемый GUI (Graphical User Interface) наиболее привычный пользователям операционных систем семейства Microsoft Windows, подразумевающих использование и манипуляцию визуальными элементами с помощью мыши и командный/консольный CLI (Command-Line Interface) подразумевающий взаимодействия с системой исключительно путем ввода текстовых команд в командной строке с клавиатуры.

В базовой установке дистрибутив OpenScaler предполагает использование исключительно CLI интерфейса с использованием интерпретатора BASH. В рамках данного курса именно данный вариант взаимодействия с системой будет рассматриваться как основной и наиболее универсальный. Тем не менее, в репозиториях OpenScaler присутствует более 5 различных графических окружений, установив которые, можно реализовать GUI интерфейс на любой вкус и цвет, но это выходит за рамки данного курса.

OpenScaler помимо локальной авторизации позволяет также осуществлять удаленное подключение к CLI интерфейсу систему посредством SSH. Для осуществления удаленного доступа до машины с установленным дистрибутивом OpenScaler необходимо использование стороннего приложения – SSH-клиента, к примеру Putty.

По умолчанию пользователю доступно шесть виртуальных консолей для взаимодействия с системой, переключение между ними осуществляется с помощью комбинации клавиш CTRL+ALT+F[1-6].

Авторизовавшись в системе по виду командной строки пользователь может легко определить уровень привилегий в системе. Учетная запись центрального администратора root обычно сопровождается командной строкой оканчивающейся на символ “#” в то время как для обычных пользователей с обычными привилегиями это “$”. К примеру, обратив внимание на рисунок 14 и вид командной строки можно определить что в системе авторизован пользователь “openeuler”, он не является администратором системы, имя системы к которой он подключен “host”, а значок “~” означает что сейчас пользователь находится в домашнем каталоге.

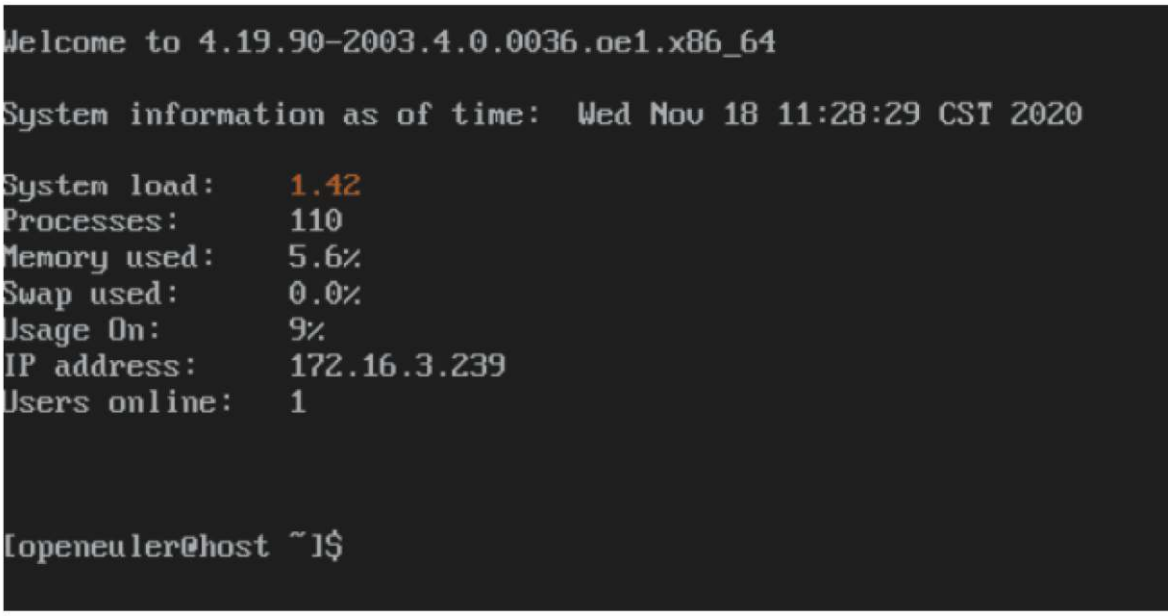

Рисунок 14. Пользователь, авторизованный в CLI интерфейсе системы

Выполните следующую команду, чтобы просмотреть информацию о системе:

cat /etc/os-release

Просмотрите информацию о системных ресурсах.

Выполните следующую команду, чтобы просмотреть информацию о ЦП:

lscpu

Выполните следующую команду, чтобы просмотреть информацию о памяти:

free

Выполните следующую команду, чтобы просмотреть информацию о диске:

fdisk -l

Выполните следующую команду, чтобы просмотреть IP-адрес:

ip addr

2.5 ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

- Что такое CLI и GUI интерфейсы и в чем их разница?
- Чем пользователь root отличается от других пользователей системы?
- В чем различия архитектур ARM и X86?
- Какие консольные инструменты и команды вы помните по итогам прочтения главы?

3. БАЗОВЫЕ КОМАНДЫ И ОПЕРАЦИИ С ИСПОЛЬЗОВАНИЕМ CLI ИНТЕРФЕЙСА

3.1 ОСНОВЫ КОМАНДНОЙ СТРОКИ

Linux невозможно представить без командной строки (терминала). Любые действия связанные с ПК вы можете выполнить прописав нужную команду. CLI интерфейс для Linux всегда играл важную роль. В отличии от других операционных систем, Linux может предложить пользователю возможность работы с ОС без графической составляющей. Такая ОС будет выглядеть как одно сплошное окно терминала, где все действия прописываются за счет различных команд. Исполняет действия так называемая оболочка или шелл (от англ. shell). Это часть ОС, которая определяет поведение командной строки и следит за выполнением команд. Ее место в иерархии представлено на рисунке 15. Существуют разные оболочки, но наиболее распространена так называемая bash, сокращение от Bourne Again Shell.

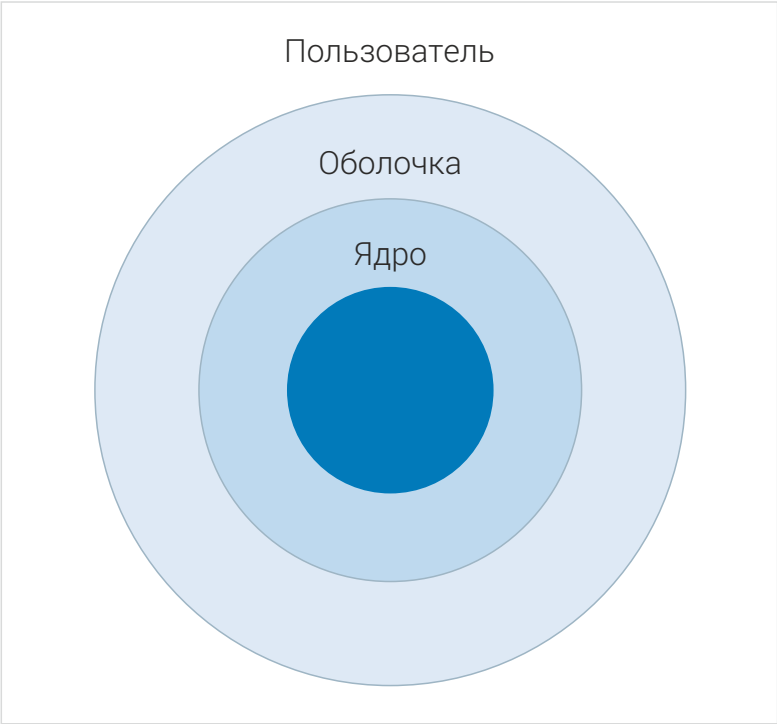

Рисунок 15. Место оболочки в ОС

По умолчанию в дистрибутиве OpenScaler отсутствует графический режим (GUI), тому есть две ключевые причины:

- Приверженность разработчиков к давнему постулату - “Клавиатура быстрее мыши” и общая более высокая эффективность CLI. К примеру, последовательность команд и действий можно автоматизировать на уровне скрипта в отличие от действий совершаемых с графическими объектами мышью.
- "Высокая стоимость красоты" - графический режим вне зависимости от выбранного пользователем рабочего окружения (DE - Desktop Environment) потребляет достаточно большое количество драгоценных ресурсов системы, которые можно использовать для других, более продуктивных задач.

В рамках данного курса все управление и настройка операционной системы будет осуществляться исключительно через CLI интерфейс. Для любого Linux дистрибутива типовым является следующий формат команд:

Команда [-опция] [аргумент]

Все команды, запускаемые из shell, делятся на внешние и внутренние. Для выполнения внешних команд системе требуется знать полный путь до директории в которой находится исполняемый файл с исполняемой указанной командой. Любая команда в Linux состоит из имени запускаемой программы (команды), опций и аргументов, передаваемых программе. Опции и аргументы отделяются от имени команды и друг от друга пробелами или знаками табуляции.

Если при запуске команды не указывается путь к исполняемому файлу, то ОС последовательно просматривает директории, содержащиеся в переменной PATH. Если в какой-нибудь из этих директорий есть программа таким именем, то она запускается на выполнение. Следует отметить еще тот факт, что ОС не ищет запускаемую программу в текущей директории.

Пример 1:

1. Запуск программы с указанием полного пути

```
$ pwd
/usr/
$ /usr/local/bin/myprog
```

2. Рапуск программы из вышестоящей директории

```
$ pwd
/usr/etc
$ ../local/bin/myprog
```

3. Запуск программы из текущей директории

```
$ pwd
/usr/local/bin
$ ./myprog
```

Команде может быть передано несколько опций, при необходимости с указанием аргументов для каждой.

ПРИМЕЧАНИЕ

Иногда бывает удобно использовать маленькую хитрость - клавишу TAB. В BASH нажатие данной кнопки позволяет автоматически дописать название команды или файла в каталоге что упрощает процесс написания сложных команд и сокращает вероятность ошибки.

В дополнение к вышесказанному, ниже в таблице 2 приведены наиболее часто используемые комбинации клавиш, знание которых может сильно облегчить жизнь пользователю.

Таблица 2. Ключевые комбинации клавиш при работе с BASH

Компонент	Минимальные требования
Стрелка вверх (Up arrow)	Покажет предыдущие выполненные пользователем команды - одно нажатие последняя выполненная команда, второе нажатие - предпоследняя и так далее.
Стрелка вниз (Down arrow)	Перемещение по истории выполненных команд (после инициации стрелкой вверх)
Home	Переместить курсор к началу строки
CTRL+A	Переместить курсор к началу строки
CTRL+E	Переместить курсор к концу строки
CTRL+C	Прекратить выполнение текущей программы
CTRL+L	Очистить экран

3.2 БАЗОВЫЕ КОМАНДЫ

Все ключевые команды в ОС Linux разбиты на следующие категории, представленные в таблице 3.

Таблица 3. Основные команды и категории

Категория команд	Ключевые команды
Авторизация и управление питанием	login, shutdown, halt, reboot, install, exit, last и другие
Работа с файлами	file, mkdir, grep, dd, find, mv, ls, diff, cp, cat, ln и другие
Управление системой	df, top, free, quota, at, ip, kill, crontab
Управление сетевым стеком	ifconfig, ip, ping, netstat, telnet, ftp, route, rlogin, finger, nslookup, ethtool
Управление файловыми системами и подсистемой хранения данных	fdisk,df, parted, mkfs, pvcreate, vgcreate, lvcreate, vgs, lvextend, mount
Управление подсистемой безопасности	passwd, su, umask, chgrp, chmod, chown, chattr, sudo, ps, who
Другие	tar, unzip, mtools, man

3.2.1 Команды входа в систему

Login

Login - это первое что вы увидите после загрузки операционной системы. Введите имя пользователя и следом его пароль, которые были заданы вами при первоначальной установке системы. По соображениям безопасности вводимые символы в строке пароля не отображаются.

Linux является многопользовательской операционной системой, поддерживающей одновременно несколько логинов одного пользователя. После прохождения авторизации в консоли, вы можете переключиться на следующую, применив комбинацию клавиш ALT+F[1-6]. К примеру, авторизовавшись на одной консоли и нажав ALT+F2 вы переключитесь на

другую, в которой также сможете авторизоваться под этим же или другим пользователем. Переключиться обратно на первую мы можете, нажав ALT+F1. Это полезно к примеру в случае использования одной консоли для запуска приложения, а второй - для отслеживания журнала ее работы.

Last

Команда “last” используется для отображения информации об авторизации пользователей в системе а также используемых ими терминалах. Просматривая вывод команды, системный администратор может определить кто пытался авторизоваться в системе. Ключевыми опциями для данной команды являются:

- **-a** отобразить имена хостов в последней колонке
- **-d** перевести IP-адрес обратно в имя хоста
- **-n** количество отображаемых строк
- **-R**, --nohostname don't display the hostname field
- **-s** отображать строки начиная с указанного времени
- **-t** отображать строки до указанного времени

exit

Команда выхода пользователя из текущей оболочки.

logout

Команда выхода пользователя из системы на данной виртуальной консоли. По завершении будет предложено повторно авторизоваться.

Примеры использования:
Авторизуйтесь в системе под пользователем root, как представлено на рисунке 16.

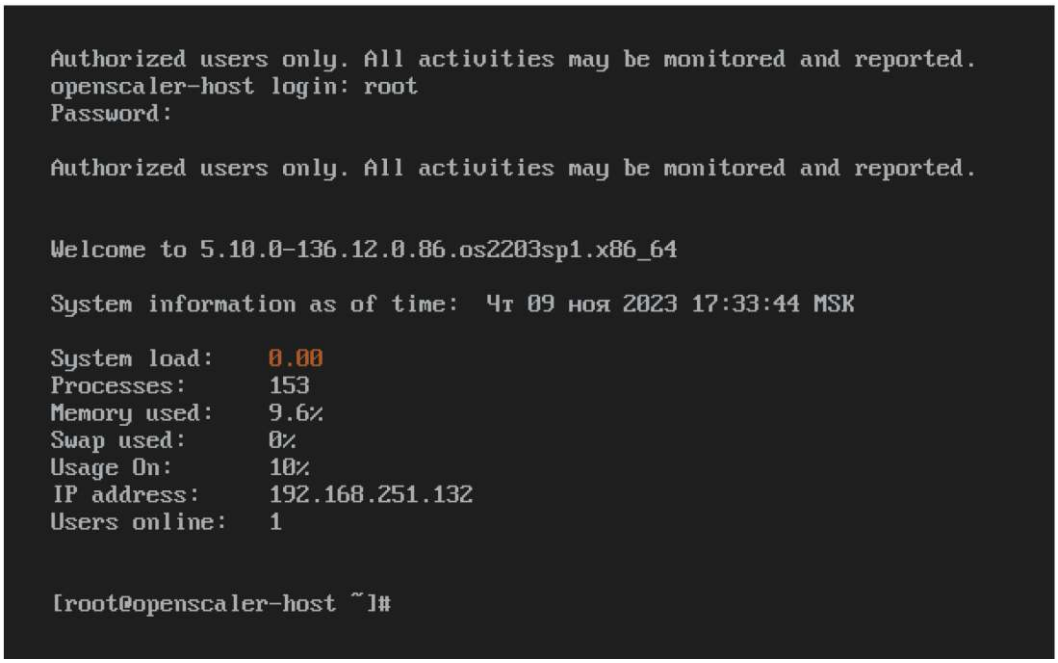

Рисунок 16. Авторизация в системе с правами пользователя root

Переключитесь на другую виртуальную консоль нажав ALT+F2. Авторизуйтесь.
Переключитесь на обычного пользователя созданного при установки системы выполнив команду **[root@host -]# su openScaler**

Выполните выход из системы **[openScaler@host -]$ exit**
Посмотрите информацию о последних авторизациях в системе выполнив команду **[root@host -]# last root**

3.2.2 Команды управления питанием системы

shutdown

Команда shutdown предназначена для выключения и перезагрузки компьютера, используя командную строку Linux. shutdown также позволяет перезагрузить или выключить компьютер в заданное время или через заданный интервал времени. Данная команда доступна только суперпользователю/администратору (root) и по умолчанию недоступна для выполнения обычными пользователями системы.

Команда имеет следующий синтаксис

Shutdown [ОПЦИИ] [ВРЕМЯ] [СООБЩЕНИЕ]

Первый аргумент [ОПЦИИ] может быть строкой времени. После аргумента вы можете при желании ввести сообщение, которое будет выведено в терминал, чтобы уведомить вошедших в систему пользователей перед выключением.

Формат строки времени – чч:мм (часы/минуты) в 24-часовом диапазоне. Указанные вами цифры определят время выполнения команды shutdown. В качестве альтернативы вы можете использовать параметр +м, где м означает минуты. Замените букву на желаемое количество минут, по истечению которых ваша система будет выключена.

Вы также можете использовать в синтаксисе now. Это будет означать то же самое, что +0 и приведёт к немедленному отключению вашего VPS. Если вы не укажете аргумент [время], Linux автоматически применит +1.

Имейте в виду, что параметр времени является обязательным, если вы хотите вывести сообщение о выключении в терминал. Файл /run/nologin будет создан за 5 минут до выключения системы, чтобы убедиться, что дальнейшие входы не будут разрешены, но только если вы указали аргумент времени.

Из других полезных опций данной команды стоит отметить следующие:

- **-H (-h)** - отключить питание
- **-r** - осуществить перезагрузку системы

Использование команды также позволяет заблаговременно оповестить всех пользователей работающих в системе о планируемой перезагрузке или выключении.

halt

Команда может быть использована только суперпользователем системы и используется для выключения системы. Реализация функции, используемой посредством данной команды, зависит от архитектуры центрального процессора; например, в архитектуре x86 для этой цели предусмотрена специальная инструкция HLT. В текущее время эта команда является устаревшей, поэтому после перехода большинства дистрибутивов на менеджер инициализации systemd, она была реализована в виде простой символьной ссылки на утилиту systemctl из состава systemd. Базовый синтаксис команды выглядит следующим образом:

 # halt [опции]

Наиболее важными параметрами являются параметры **--poweroff** и **--reboot**, позволяющие выключить и перезагрузить систему соответственно, параметр **--force**, предназначенный для принудительного прекращения работы центрального процессора путем осуществления системного вызова без подготовки системы, параметр **--wtmp-only**, позволяющий добавить запись в файл журнала /var/log/wtmp и не прекращать работу центрального процессора, параметр **--no-wtmp**, позволяющий прекратить работу центрального процессора без добавления записи в файл журнала /var/log/wtmp, а также параметр **--no-wall**, позволяющий не отправлять сообщение о прекращении работы центрального процессора всем пользователям системы.

reboot

Команда служит для осуществления перезагрузки системы и может быть использована только суперпользователем системы (root). Из ключевых параметров стоит отменить:

- **-f** - осуществить незамедлительную перезагрузку системы
- **-w** - не осуществлять перезагрузку но оставить запись в журнале /var/log/wtmp

3.2.3 Иерархия директорий файловой системы

При работе с любым дистрибутивом Linux нужно держать в голове концепцию, которая лежит в основе системы: “Всё - это файл”. Расшифровывается просто: любые данные и процессы операционной системы можно выразить как поток байтов, занимающий определённый объём файловой системы на дисковом накопителе.

Соответственно, всё, что занимает место, можно назвать файлом. А уж файлы можно структурировать как душе угодно:

- обычные файлы
- текстовые
- бинарные
- файлы изображений
- архивы и т.п.
- специальные
- блочные для обозначения устройств (b)
- символьные (c)
- ссылочные для представления символьных ссылок (l)
- файлы сокетов для связи между разными процессами и пр. (s)
- директории — файлы, в которых хранятся другие файлы (d)

Узнать тип файла можно с помощью команды **ls -l**

Первый символ в каждой строке вывода обозначает тип файла. Уточнить его можно с помощью команды **file [имя]**, которой в качестве входных данных передаётся имя файла/папки.

Linux-системы имеют много общего, и файловая структура — одна из этих общих черт. Знакомство с ней поможет увереннее ориентироваться в операционной системе в целом. Давайте погрузимся немного поглубже и посмотрим, что лежит в корне Linux-системы.

Собственно, в корне, или корневой директории “/”, лежат все данные сервера, распределённые по разным каталогам. При этом каждый каталог имеет своё значение и содержит файлы, сгруппированные по определённому критерию. По умолчанию структура директорий состоит из следующих папок представленных в таблице 4.

Таблица 4. Назначение каталогов в корневой файловой системе

Имя директории	Назначение
/bin	Директория /bin содержит исполняемые бинарные файлы различных служб, доступные для запуска любым пользователям сервера. Включает и исполняемые файлы базовых команд, которые мы ранее перечисляли в таблице 3, к примеру echo, cat, cd, pwd, ls и пр.
/boot	Содержимое этого каталога содержит сердце системы — файлы загрузчика и ядра. Без этой директории операционная система не сможет запуститься.
/dev	Эта директория содержит файлы устройств, подключенных к серверу. Включает терминал, через который мы отдаём серверу команды (tty*), а также специальные устройства типа null, random, zero
/etc	Эта директория по умолчанию содержит все конфигурационные файлы служб, а в некоторых случаях — и скрипты для их запуска и отключения.
/home	Эта папка предназначена для домашних каталогов пользователей. Каждый раз, когда вы создаёте локального пользователя без указания домашней папки, здесь автоматически создаётся одноимённая папка — его домашняя директория по умолчанию.
/lib и /lib64	В директории /lib хранятся библиотеки, необходимые для работы системных служб, файлы которых размещены в папках /bin и /sbin. А также данные для загрузки системы и модуля ядра ОС. Директория /lib64 предназначена для тех же целей, только включает в себя библиотеки для 64-х битных систем.
/mnt	Используется для временного монтирования внешних устройств.
/opt	В эту папку обычно устанавливается пользовательское программное обеспечение. Собственно, это и заложено в название — директория для опциональных, необязательных данных.

Имя директории	Назначение
/proc	В этом каталоге хранятся процессы и системная информация ОС, представленная в виде файлов, так называемая “виртуальная файловая система”. Все данные в этом разделе генерируются автоматически и обновляются на лету. Все директории с наименованием в цифровом формате содержат информацию о запущенных процессах. В частности, название папки соответствует PID, идентификатору процесса.
/root	Домашняя папка для суперпользователя root
/sbin	Как и /bin, содержит исполняемые бинарные файлы системных служб. Но, в отличие от /bin, только служб, запуск которых возможен исключительно с правами администратора.
/srv	Эта папка предназначена для сервисных нужд — чтобы пользователи сервера могли найти какие-то общедоступные данные для конкретной службы, например, веб-сервера, FTP-сервера. По умолчанию не содержит данных.
/tmp	Эта папка используется операционной системой и различными внутренними службами для хранения временных файлов.
/usr	В этом каталоге хранятся исполняемые файлы, библиотеки и файлы документации (man) для внутренних служб, компоненты ядра для функционирования операционной системы, а также данные программ, установленных пользователями.
/var	Директория /var содержит часто изменяемые данные. Например, кэши (/var/cache), логи (/var/log), очереди (/var/spool).
/run	Этот каталог включает в себя данные, обрабатываемые и хранимые в оперативной памяти — например, PID процессов, информацию о ходе их выполнения, активные сокеты и пр. Так называемая временная файловая система. Она сбрасывается при каждой перезагрузке сервера.

3.2.4 Пути файлов и работа с ними

Все файлы в Linux имеют определенный адрес в файловой системе, с помощью которого мы можем получить к ним доступ с помощью файлового менеджера или консольных утилит.

В операционной системе Linux может быть несколько видов путей к файлу:

- Полный, абсолютный путь от корня файловой системы - он начинается от корня “/” и описывает весь путь к файлу;
- Относительный путь - это путь к файлу относительно текущей папки.

К примеру, в системе существует две директории: /home/smc/config и /home/smc/bin, если пользователь находится в каталоге /home/smc/config directory а нужно перейти в каталог /home/smc/bin - то можно это сделать одним из следующих вариантов:

- Используя абсолютный путь: cd /home/smc/bin
- Используя относительный путь: cd ../bin., где “.” означает директорию уровнем выше.

3.2.5 Базовые команды по работе с файлами

ls

Команда ls в Linux используется для вывода содержимого текущей рабочей директории. По умолчанию, команда ls отображает имена файлов и поддиректорий в текущей директории в алфавитном порядке.

Базовый синтаксис команды следующий: **ls [опции] [имя файла или директории]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-l:** отображает длинный формат вывода, который включает дополнительную информацию о каждом файле, такую как права доступа, владелец, группу, размер, дату создания и т.д.
- **-a:** отображает все файлы, включая скрытые файлы, начинающиеся с точки.
- **-t:** отображает файлы в порядке последней модификации, начиная с самого недавнего.
- **-r:** отображает файлы в обратном порядке.

cd

Команда используется для смены текущего рабочего каталога пользователя и обладает следующим синтаксисом: **cd [имя директории]**

К примеру:

- **cd /usr** - перейти в директорию /usr
- **cd** - перейти в домашний каталог пользователя
- **cd ../** - переход в директорию на уровень выше

pwd

Команда pwd выводит текущую рабочую директорию. При вызове pwd выводится полный путь к текущей рабочей директории.

Команда pwd принимает только две опции:

- **-L (--logical)** — не разрешать симлинки.
- **-P (--physical)** — отображать физическую директорию без симлинков.

Поупражняемся в использовании команд ls, cd, pwd:

1. Проверим содержимое домашнего каталога пользователя root
   a. **[root@host -]# ls /**

2. Перейдем в корневую директорию файловой системы
   a. **[root@host -]# cd /**

3. Посмотрим информацию для всех файлов в корневой директории и отсортируем их в хронологическом порядке
   a. **[root@host /]# ls –alt**

4. Используя абсолютный путь перейдем в директорию /home
   a. **[root@host /]# cd /home**

5. Используя относительный путь перейдем во вложенную директорию в каталоге home (одноименная директория для пользователя созданная на этапе установки системы), к примеру openscaler
   a. **[root@host home]# cd openscaler**

6. Проверим в какой конкретно директории мы сейчас находимся
   a. **[root@host openscaler]# pwd**

7. Перейдем в директорию уровнем выше
   a. **[root@host openscaler]# cd ../**

8. Вернемся в домашний каталог нашего пользователя
   a. **[root@host home]# cd**

mkdir

Команда mkdir позволяет создавать новые директории. mkdir означает «создать директорию». С помощью mkdir можно также устанавливать разрешения, создавать несколько директорий, в том числе вложенных (для этого используется опция -p) одновременно и многое другое.

Базовый синтаксис команды следующий:

mkdir [опции] имя_директории

Примеры наиболее распространенных случаев использования команды:

1. Создать директорию “dir1”
   a. **[root@host -]# mkdir dir1**

2. Создать несколько директорий, к примеру, dir1 и dir2
   a. **[root@host -]# mkdir dir1 dir2**

3. Создать директорию dir1 и две вложенные друг в друга поддиректории
   a. **[root@host -]# mkdir -p dir1/dir2/dir3**

4. Создать директорию и показать результат работы
   a. **[root@host -]# mkdir -pv dir1/dir2**

touch

Команда служит для создания нового пустого текстового файла и обладает следующим базовым синтаксисом: **touch [опции] имя_файла**

Примеры наиболее распространенных случаев использования команды:

1. Создать новый пустой текстовый файл
   a. **[root@host -]# touch file**

2. Изменить время последнего доступа к файлу
   a. **[root@host -]# touch -a file**

3. Изменить время последнего изменения файла
   a. **[root@host -]# touch -m file**

cp

Команда используется для копирования файлов и директорий и обладает следующим базовым синтаксисом:

cp [опции] /путь/к/файлу/источнику /путь/к/файлу/назначения

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-a** - режим резервного копирования, при котором сохраняются все атрибуты, ссылки
- **-r,** --recursive - копировать папку рекурсивно;
- **-l,** --link - создавать жесткие ссылки вместо копирования.

Примеры наиболее распространенных случаев использования команды:

1. Создать файл f2 являющийся копией файла f1
   a. **[root@host -]# cp f1 f2**

2. Копировать файл f1 в директорию d1 с сохранением имени
   a. **[root@host -]# cp f1 d1/**

3. Копировать несколько файлов в одну директорию
   a. **[root@host -]# cp f1 f2 f3 d1/**

4. Если f2 существует - задать вопрос, нужно ли его переписать
   a. **[root@host -]# cp -i f1 f2**

5. Копировать директорию d1 включая все ее файлы и поддиректории в d2
   a. **[root@host -]# cp -r d1 d2**

6. Копировать файл f1 с сохранением всех его атрибутов
   a. **[root@host -]# cp -a f1 f2**

mv

Команда mv (move) используется для перемещения и переименования файлов из каталогов и доступна во всех дистрибутивах Linux. При перемещении файла или каталога в новый каталог сохраняется базовое имя файла. При перемещении файла все ссылки на другие файлы сохраняются.

Обладает следующим базовым синтаксисом:

mv [опции] имя_источника имя_назначения

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-b** - Создавать резервные копии целевых файлов;
- **-f** - Переписывать существующие файлы не спрашивая;
- **-i** - Спрашивать перед тем как переписывать;
- **-u** - Перемещать только если источник новее чем файл назначения, или когда файл назначения отсутствует.

Примеры наиболее распространенных случаев использования команды:

1. Переместить файл f1 в f2
   a. **[root@host -]# mv f1 f2**

2. Переместить файл f1 в директорию d1 с сохранением имени
   a. **[root@host -]# mv f1 d1/**

3. Переместить несколько файлов в одну директорию
   a. **[root@host -]# mv f1 f2 f3 d1/**

4. Если f2 существует - задать вопрос, нужно ли его переписать
   a. **[root@host -]# mv -i f1 f2**

5. Переместить f1 в существующий f2 предварительно сделав резервную копию последнего
   a. **[root@host -]# mv –b f1 f2**

6. Переместить директорию f1 со всеми поддиректориями и файлами в f2
   a. **[root@host -]# mv -r d1 d2**

7. Переместить все содержимое d1 в d2 и отобразить результат работы команды
   a. **[root@host -]# mv -rv d1 d2**

8. Переместить все содержимое d1 в d2 перезаписав последнюю
   a. **[root@host -]# mv -f f1 f2**

rm

Команда rm (remove) используется для удаления файлов. Они удаляются навсегда, поэтому следует соблюдать осторожность и желательно иметь резервные копии.

Обладает следующим базовым синтаксисом:

rm [опции]… [имя_файла]…

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-r** - Рекурсивно удалять каталоги и их содержимое;
- **-i** - Запрашивать подтверждение перед каждым удалением.

Примеры наиболее распространенных случаев использования команды:

1. Удалить файл f1
   a. **[root@host -]# rm f1**

2. Удалить несколько файлов
   a. **[root@host -]# rm f1 f2**

3. Удалить файл f1 с дополнительным подтверждением пользователя
   a. **[root@host -]# rm -i f1**

4. Удалить директорию d1 и все ее содержимое
   a. **[root@host -]# rm -r d1**

5. Удалить директорию d1 и все ее содержимое и показать результат работы команды
   a. **[root@host -]# rm –rv d1**

find

Команда позволяет эффективно находить файлы по различным признакам, таким как имена, типы, размеры и время модификации

Обладает следующим базовым синтаксисом: **find [путь] [выражение]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-name** - поиск по имени файла;
- **-perm** - поиск по правам доступа к файлу;
- **-user** - поиск по принадлежности файла пользователю;
- **-mtime** - поиск по времени изменения файла.

Примеры наиболее распространенных случаев использования команды:

1. Найти файл, в чьем названии есть “book”
   a. **[root@host -]# find -name "*book*"**

2. Найти файлы ,чей размер равен нулю
   a. **[root@host -]# find -size 0**

3. Найти файлы типа soft-link
   a. **[root@host -]# find -type l**

4. Найти файлы в каталоге /etc, в чьем имени есть "passwd”
   a. **[root@host -]# find /etc -name "*passwd"**

5. Найти пустые файлы и каталоги и удалить их
   a. **[root@host -]# find -empty -delete**

locate

Команда locate используется для поиска файлов, расположенных на машине пользователя или на сервере. Фактически она выполняет ту же работу, что и команда find, однако, ведёт поиск в собственной базе данных. Find же шаг за шагом проходит через всю иерархию директорий.

Обладает следующим базовым синтаксисом: **locate [опции] [выражение]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-e** - поиск файлов, существующих на момент запуска команды.
- **-r** - поиск с использованием регулярных выражений
- **-d** - меняет базу данных для поиска, установленную по умолчанию, на пользовательскую.

При запуске команды происходит поиск по базе данных индексов. В случае если база данных не существует или давно не обновлялась возможно возникновение ошибки **locate: can not open `/var/lib/mlocate/mlocate.db': No such file or directory"** .
Для ее устранения выполните команду **updatedb**

which

Команда используется для определения наличия исполняемого файла или команды в местах расположения указанных в переменной PATH.

Обладает следующим базовым синтаксисом: **which [опции] имя_исполняемого_файла**

Примеры наиболее распространенных случаев использования команды:

1. Найти абсолютный пусть для исполняемого файла ls
   a. **[root@host -]# which ls**

2. Одновременный поиск нескольких файлов
   a. **[root@host -]# which cp mv rm**

3.2.5.1 Символические и жесткие ссылки

ln

Команда ln используется для создания жестких (hard) или символических (soft) ссылок на файлы или директории. Жесткая ссылка создает новое имя для файла или директории, указывая на тот же индексный узел (inode) в файловой системе. Символическая ссылка создает новый файл, который содержит путь к оригинальному файлу или директории.

Ссылки в Linux бывают двух типов: символические и жесткие. Не смотря на то, что оба типа называются ссылками, они имеют существенные отличия друг от друга.

Символическая ссылка (symbolic link) — это специальный файл, который является ссылкой на другой файл или каталог (их еще называют целевым файлом, целевым каталогом). Символические ссылки также называют символьными, мягкими ссылками (soft links) или сим-ссылками (sym-link). Символическая ссылка не содержит в себе внутри копии самого файла, на которую она указывает. Она является всего лишь указателем на файл. Не смотря на это, символическая ссылка обладает собственными правами доступа, так как сама является небольшим файлом, который содержит путь до целевого файла. Связь между символической ссылкой и файлом, на который она указывает, является “мягкой”. Если удалить символическую ссылку, то файл, на который она указывает, не удаляется. Если удалить файл, на который указывает ссылка, то сама ссылка не обновляется и остается на диске. При этом она указывает на уже несуществующий файл. Аналогично, если переименовать или переместить целевой файл, то ссылка не обновляется автоматически. При создании символических ссылок можно указывать относительный путь до целевого файла. В таком случае ссылка считает, что относительный путь указан относительно каталога, в котором создана сама ссылка (но не относительно каталога, из которого она была создана).

Жесткая ссылка (hard link) является своего рода синонимом для существующего файла. Когда пользователь создает жесткую ссылку, создается дополнительный указатель на существующий файл, но не копия файла. Жесткие ссылки выглядят в файловой структуре как еще один файл. Если пользователь создает жесткую ссылку в том же каталоге, где находится целевой файл, то они должны иметь разные имена. Жесткая ссылка на файл должна находится в той же файловой системе, где и другие жесткие ссылки на этот файл. В любом дистрибутиве Linux каждый файл имеет уникальный идентификатор - индексный дескриптор (inode). Это число, которое однозначно идентифицирует файл в файловой системе. Жесткая ссылка и файл, для которой она создавалась имеют одинаковые inode. Поэтому жесткая ссылка имеет те же права доступа, владельца и время последней модификации, что и целевой файл. Различаются только имена файлов. Фактически жесткая ссылка это еще одно имя для файла. Жесткие ссылки нельзя создавать для директорий. Жесткая ссылка не может указывать на несуществующий файл.

Из всего вышеописанного важно запомнить ключевые отличия мягких и жестких ссылок друг от друга:

Символическая ссылка:

- Указывает на целевой файл или каталог. Фактически является небольшим файлом, содержащим путь до целевого файла.
- Не содержит внутри себя содержимого самого файла. Содержит путь к целевому файлу.
- Имеет собственные права доступа, которые не распространяются на целевой файл.

- Удаление / переименование / перемещение целевого файла не обновляет автоматически ссылку. Ссылка начинает указывать на несуществующий файл, становится неработающей.
- Изменение прав доступа у целевого файла не обновляет права доступа у ссылки.
- Может быть создана для директории.
- Ссылка и целевой файл имеют разные файловые индексы (inode) в файловой системе.
- Может указывать на несуществующий файл.
- Символическая ссылка может использовать относительный путь до целевого файла.

Жесткая ссылка:

- Является своего рода еще одним именем на файл.
- Не может указывать на директорию.
- Нельзя создавать жесткие ссылки между файлами разных файловых систем.
- Не может указывать на несуществующий файл.
- Жесткая ссылка и файл, для которого она создавалась, имеют одинаковые индексы (inode) в файловой системе.

Но вернемся к команде ln и работе со ссылками с ее использованием. Команда обладает следующим базовым синтаксисом:

ln [опции] имя_источника имя_директории_или_файла

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-b** - удалить существующую ссылку и создать новую
- **-f** - принудительное исполнение без дополнительных уведомлений
- **-i** - интерактивный режим, если ссылка уже существует пользователь будет проинформирован
- **-n** - считать символические ссылки обычными директориями
- **-s** - создать символическую ссылку, она же soft link

Примеры наиболее распространенных случаев использования команды:

1. Создать жесткую ссылку на файл f1 и назвать ее f2
   a. **[root@host -]# ln f1 f2**

2. Создать символьную ссылку на файл f1 и назвать ее f2
   a. **[root@host -]# ln -s f1 f2**

3. Создать символьную ссылку на файл f1 и назвать ее f2 но в интерактивном режиме, если ссылка уже существует - пользователь будет уведомлен
   a. **[root@host -]# ln –i f1 f2**

4. При создании ссылки f2 на файл f1 - если таковая уже существует - она будет пересоздана
   a. **[root@host -]# ln –b f1 f2**

3.2.6 Базовые команды для работы с архивами

gzip

Команда gzip предназначена для сжатия данных без потерь с помощью одноименной утилиты, использующей алгоритм Лемпела-Зива (LZ77) с кодированием Хаффмана. Целью использования данной утилиты является экономия дискового пространства. Данный алгоритм является стандартным алгоритмом утилиты zip и используется по умолчанию в архивах формата ZIP. По статистическим данным, в среднем процент сжатия архивируемых данных составляет 60%-70%.

Команда обладает следующим базовым синтаксисом: **gzip [опции] имя_дир/файла**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-d** - Распаковать архивный файл
- **-f** - принудительно создать архив даже если в него попадают символьные ссылки и есть конфликты имен
- **-r** - Рекурсивно добавить в архив все файлы и поддиректории указанной директории
- **-l** - Посмотреть информацию об указанном архивном файле
- **-v** - Информировать пользователя о процессе выполнения команды во время ее деятельности

Примеры наиболее распространенных случаев использования команды:

1. Создать архив с файлом f1
   a. **[root@host -]# gzip f1**

2. Создать рекурсивно архивы всех файлов внутри директории d1 и её поддиректорий
   a. **[root@host -]# gzip -r d1**

3. Распаковать архив f1
   a. **[root@host -]# gzip -d f1**

4. Распаковать архив d1 и продемонстрировать ход выполнения работы
   a. **[root@host -]# gzip –drv d1**

tar

Tar — одна из наиболее широко используемых команд Linux для сжатия. Tar расшифровывается как “Tape archive” и используется для сжатия файлов и папок. В большинстве случаев результатом сжатия с использованием утилиты tar является файл с расширением .tar. Последующие сжатия выполняются с помощью gzip, в результате получаем файл *.tar.gz. С помощью tar пользователь может сжимать и распаковывать архивные файлы.

Команда обладает следующим базовым синтаксисом: **tar [опции] имя_дир/файла**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-c** - Создать новый архив
- **-x** - Извлечь файлы из архива
- **-t** - Показать список содержимого архива
- **-z** - Перенаправить вывод в архиватор gzip
- **-v** - Информировать пользователя о процессе выполнения команды во время ее деятельности
- **-j** - Перенаправить вывод в архиватор bzip2

Примеры наиболее распространенных случаев использования команды:

1. Упаковать в архив всю директорию dir1 и ее содержимое
   a. **[root@host -]# tar -cf ball.tar dir1**

2. Показать содержимое архива
   a. **[root@host -]# tar -tf ball.tar**

3. Распаковать архив в текущую директорию
   a. **[root@host -]# tar -xf ball.tar**

4. Упаковать в архив и сжать с помощью gzip
   a. **[root@host -]# tar -czf ball.tar.gz dir1**

5. Упаковать в архив и сжать с помощью bzip2
   a. **[root@host -]# tar -cjf ball.tar.bz2 dir1**

6. Распаковать архив в каталог /tmp
   a. **[root@host -]# tar -xf ball.tar -C /tmp**

7. Показать во время работы процесс распаковки файлов
   a. **[root@host -]# tar -xvf ball.tar**

3.2.7 Команды для работы со встроенной справочной системой

man

Команда man, предназначена для форматирования и вывода справочных страниц. Каждая страница справки является самостоятельным документом и пишется разработчиками соответствующего программного обеспечения. Является основой справочной системы.

Все справочные материалы включенные в базу данных справочных материалов man разделены на девять категорий представленных в таблице 5.

Таблица 5. Категории справочных материалов man

Номер категории	Тип контента
1	Инструкции и программы, исполняемые в оболочке
2	Системные вызовы и функции ядра ОС
3	Функции, реализуемые в библиотеках
4	Специальные файлы (обычно из каталога /dev)
5	Форматы файлов
6	Игры
7	Контент, не подпадающий под другие категории
8	Команды системного администрирования (обычно доступные только root)
9	Специфические функции ядра ОС

Основная наиболее используемая информация и справочные страницы находятся в категориях 1, 4, 5 и 8 и поиск данных происходит именно в них. К примеру, если выполнить команду “**man sleep**” мы увидим справку по использованию данной программы, но если нам требуется получить специфическую информацию о системных вызовах и/или используемых ею библиотеках то запрос должен быть “**man 3 sleep**”.

Документацию в man можно искать также по ключевым словам, к примеру
“man -k ключевое_слово”

Вывод команды **[root@host -]# man -k sleep** представлен на рисунке 17.

```
[root@openscaler-host ~]# man -k sleep
sleep.conf.d (5)     - Suspend and hibernation configuration file
systemd-hibernate.service (8) - System sleep state logic
systemd-hybrid-sleep.service (8) - System sleep state logic
systemd-sleep (8)    - System sleep state logic
systemd-sleep.conf (5) - Suspend and hibernation configuration file
systemd-suspend-then-hibernate.service (8) - System sleep state logic
systemd-suspend.service (8) - System sleep state logic
usleep (1)           - sleep some number of microseconds
[root@openscaler-host ~]#
```

Рисунок 17. Результат выполнения поиска по справочной системе man

help

Конечно же, невозможно запомнить все имеющиеся флаги для различных команд. Для этого существует встроенное руководство по каждой команде, которое можно вызвать специальным флагом -h или командой help.

Команда обладает следующим базовым синтаксисом:
help [опции] [команда]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-d** - Представляет базовое описание программы или команды
- **-s** - Предоставляет описание синтаксиса программы или команды

Примеры наиболее распространенных случаев использования команды:

1. Получить информацию по команде pwd: ее описание, синтаксис
   a. **[root@host -]# help pwd**
   b. **[root@host -]# help –d pwd**
   c. **[root@host -]# help –s pwd**

Результат исполнения этих команд представлен на рисунке 18.

```
[root@openscaler-host ~]# help pwd
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
                directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.
[root@openscaler-host ~]# help -d pwd
pwd - Print the name of the current working directory.
[root@openscaler-host ~]# help -s pwd
```

Рисунок 18. Результат получения справочных данных по команде pwd

3.3 Вопросы для самопроверки

- Каков результат исполнения команды “cd ~”?
- В чем разница между командами “halt” и “shutdown”?
- В чем разница между относительным и абсолютным путем?
- В чем разница между жесткими и символьными ссылками?
- В чем различается поведение жестких и символьных ссылок при удалении исходного файла на который они должны ссылаться?

4. РАБОТА С КОНСОЛЬНЫМИ ТЕКСТОВЫМИ РЕДАКТОРАМИ И ТЕКСТОМ

4.1 НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫЕ ТЕКСТОВЫЕ РЕДАКТОРЫ

Работа с текстом в CLI интерфейсе является основой для пользователя, поскольку настройка разного рода служб и сервисов операционной системы зачастую также представляет из себя редактирование конфигурационных файлов приложений (по сути тех же текстовых файлов). По сему знания как минимум одного из ключевых консольных текстовых редакторов является обязательным.
На сегодняшний день широко распространены следующие редакторы: emacs, nano, vi, vim.

emacs

Emacs — один из наиболее мощных и широко распространённых редакторов, используемых в мире Unix/Linux. По популярности он соперничает с редактором vi и его клонами.

В зависимости от ситуации, Emacs может быть:

- текстовым редактором;
- программой для чтения почты и новостей Usenet;
- интегрированной средой разработки (IDE);
- операционной системой;
- всем, чем угодно.

Всё это разнообразие достигается благодаря архитектуре Emacs, которая позволяет расширять возможности редактора при помощи языка Emacs Lisp. На языке C написаны лишь самые базовые и низкоуровневые части Emacs, включая полнофункциональный интерпретатор языка Lisp. Таким образом, Emacs имеет встроенный язык программирования, который может использоваться для настройки, расширения и изменения поведения редактора. В действительности, большая часть того редактора, с которым пользователи Emacs работают в наши дни, написана на языке Lisp.

nano

GNU nano — небольшой и удобный текстовый редактор, который входит в стандартную сборку нашего дистрибутива. Помимо стандартных функций терминального текстового редактора nano может выполнять отмену/возврат изменений, подсвечивать синтаксис, выполнять интерактивный поиск и замену текста и многое другое. Изначально был написан как замена закрытому редактору Pico в 1999 году. К сожалению, не поддерживает в отличие от своих конкурентов возможности расширения функционала и продвинутые функции, такие как макросы и одновременное редактирование нескольких файлов.

vi

Является одним из самых старых текстовых редакторов. Этот редактор обладает несколько своеобразным интерфейсом и, поначалу, работа с ним вызывает у неопытного пользователя серьезные затруднения, но тем не менее этот редактор очень популярен и многие тысячи людей используют именно его для редактирования текстов. Практически в любой UNIX совместимой системе имеется какая-либо реализация vi.

Большая часть экрана используется для отображения редактируемого файла. Последняя строка экрана используется для ввода команд и вывода различной информации. Редактор может находиться либо в режиме редактирования, либо в режиме ввода команд. Для того, чтобы совершать какие либо действия Вы должны находиться в нужном режиме. После запуска редактор находится в командном режиме. Для перехода из режима редактирования в командный режим используется клавиша Esc. Для того, чтобы начать редактирование файла используется команда **vi имя_файла**.

На рисунке 19 представлено графическое представление режимов работы данного редактора.

Рисунок 19. Режимы работы редактора VI

Основные возможности в командном режиме

Перемещение по файлу:

- **h, left-arrow** переместить курсор влево на один символ
- **j, down-arrow** переместить курсор вниз на одну строку
- **k, up-arrow** переместить курсор вверх на одну строку
- **l, right-arrow** переместить курсор вправо на один символ
- **/text<cr>** найти строку text в файле и поместить курсор на ее первый символ. После этого можно использовать клавиши n и Shift-n для перемещения к следующему или предыдущему включению строки.

Переход в режим редактирования:

- **i** начать ввод текста перед курсором
- **a** начать ввод текста после курсора
- **o** вставить строку после текущей и начать ввод текста в ней
- **O** вставить строку перед текущей и начать ввод текста в ней

Копирование, вставка и удаление:

- **yy y$ yw** скопировать строку, строку от позиции курсора до конца, слово.
- **dd d$ dw** удалить строку, строку от позиции курсора до конца, слово.
- **x** удалить символ
- **p** вставить содержимое буфера после курсора
- **P** вставить содержимое буфера перед курсором
- **u** отменить последнюю операцию

Сохранение и чтение файлов, выход из редактора:

- **:w<cr>** сохранить файл
- **:w filename<cr>** сохранить файл под указанным именем
- **:r filename<cr>** вставить содержимое указанного файла
- **:q<cr>** выйти из редактора
- **:wq<cr>** сохранить файл и выйти из редактора
- **:q!<cr>** выйти без сохранения файла

4.2 РЕДАКТИРОВАНИЕ ТЕКСТОВЫХ ФАЙЛОВ С ПОМОЩЬЮ РЕДАКТОРА VIM

Разберемся на примере в основном функционале данного текстового редактора.

Команда имеет следующий синтаксис **vim [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-c** - выполнить указанную команду перед открытием файла
- **-R** - Открыть файл в режиме только для чтения
- **-M** - Открыть в режиме только для чтения без возможности принудительного сохранения
- **-r** - восстановить прерванную сессию

Примеры наиболее распространенных случаев использования команды:

1. Открыть в редакторе vim файл с именем newfile. Если такой файл существует то его содержимое будет представлено на экране, если не существует, то пользователь увидит отметку “New File” как представлено на рисунке 20

   a. **[root@host -]$ vim filename**

Рисунок 20. Создание нового файла в VI

2. Открыть файл в режиме только для чтения пользователь увидит отметку “filename” [readonly]”.

   a. **[root@host -]$ vim -R filename**

3. Если открыв файл в режиме только для чтения, попытаться перейти в режим редактирования, то пользователь получит уведомление -- INSERT – W10: Warning: Changing a readonly file
4. Если по завершении модификации файла в режиме только для чтения, попытаться сохранить файл (выполнив :qa), то пользователь получит на экране следующее предупреждение E45: 'readonly' option is set (add ! to override)
5. Открыть файл в режиме только для чтения без возможности насильственного сохранения и изменения. При работе в данном режиме пользователю будет выдано следующее предупреждение E21:Cannot make changes,'modifiable' is off

   a. **[root@host -]# vim –M filename**

Передвижение курсора в vim возможно с использованием ключевых горячих клавиш, упрощающих процесс работы, представленных в таблице 6.

Таблица 6. Горячие клавиши vi для перемещения курсора

Комбинация клавиш	Эффект
h или стрелка влево	Переместить курсор на один символ влево
j или стрелка вниз	Переместить курсор на одну строку вниз
k или стрелка вверх	Переместить курсор на одну строку вверх
l или стрелка вправо	Переместить курсор на один символ вправо
0	Переместить курсор на начало данной строки
$	Переместить курсор на конец данной строки
g0	Переместить курсор к началу первой строки, отображаемой на экране
g$	Переместить курсор к началу последней строки, отображаемой на экране
:n	Переместить курсор к началу n-ой строки
gg	Переместить курсор к началу документа
G	Переместить курсор к концу документа

Операции с текстовыми данными в файле такие как копирование, вставка, удаление также могут быть осуществлены посредством горячих клавиш представленных в таблице 7.

Таблица 7. Горячие клавиши для операций с данными

Тип задачи	Комбинация клавиш	Эффект
Репликация данных	yy или y	Скопировать данную строку текста
	y[n]w	Скопировать n число слов
Вставка	p над строкой	Вставить строку над строкой
	p под строкой	Вставить строку под строкой
Удаление / Вырезание	[n]dd	Удалить/вырезать n число слов
	d[n]w	Удалить/вырезать n число строк

В командном режиме vi можно ввести **:set nu** для отображения номеров строк в документе.

Также в командном режиме можно производить поиск по документу и осуществлять быструю замену/вставку нужных данных.

1. Найти строку включающую слово “word”
   a. **:/word**
   b. Нажмите n для продолжения поиска и отображения других результатов
2. Заменить “word1” на “word2” в строках 1-5.
   a. **:1,5s/word1/word2/g**
3. Заменить “word1” на “word2” во всем документе
   a. **:%s/word1/word2/gi**

Также в редакторе vi можно временно включить подсветку найденных результатов в документе с целью облегчения их поиска и чтения. Для того чтобы включить этот режим на постоянной основе, необходимо отредактировать конфигурационный файл /**etc/vimrc** добавив в него **hlsearch**.

Примеры использования:

1. Подсветить все найденные результаты при поиске “Hello”
   a. В командном режиме **:set hlsearch**
   b. Осуществить поиск **:/hello**
2. Для отключения подсветки результатов в командном режиме выполнить **:set nohlsearch**

Для осуществления модификации (внесения изменений) в файл осуществите следующую последовательность действий.

1. Откройте нужный текстовый файл в редакторе vi
   a. **[root@host -]# vim filename**
   b. Для перехода в режим ввода данных нажмите **i** внизу экрана отобразится сообщение о том, что активен режим ввода данных **-- INSERT --**
   c. Если была допущена ошибка, можно выйти из режима ввода данных нажав **ESC** и далее **u** Одно нажатие отменяет одно последнее изменение и так далее.
   d. Если вы хотите восстановить ранее отмененное внесение изменений, нажмите **CRTL+r**

После редактирования файла необходимо провести его сохранение, для этого в первую очередь надо выйти из режима редактирования, нажав на ESC, вернувшись в командный режим. Выполните одно из следующих действий:

- **:w** - Сохранить файл
- **:wq** - Сохранить файл и выйти из редактора
- **:q!** - Принудительно выйти из редактора и не сохранять изменений
- **:q** - Выйти из редактора (только если не были внесены изменения в документ)
- **:wq!** - Принудительно сохранить и выйти из редактора

4.3 КОМАНДЫ ДЛЯ РАБОТЫ С ТЕКСТОВЫМИ ДАННЫМИ

В таблице 8 представлены ключевые команды, используемые для работы с текстовыми данными в CLI.

Таблица 8. Ключевые команды для работы с текстовыми данными в CLI

Тип действий	Команды
Просмотр файла	cat, more, less
Выдержки из файла	head, tail
Извлечение данных из файла	cut, awk, grep
Сортировка и сравнение файлов	wc, sort, diff
Инструментарий работы с текстом	sed, tr

4.3.1 Вывод/просмотр текстовых данных

cat

Команда cat, являясь сокращением английского слова «concatenate» (конкатенация), позволяет создавать, объединять, а также выводить содержимое файлов в командной строке или в другом файле.

Команда имеет следующий синтаксис

cat [опции] [имя_файла]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-n** - пронумеровывать все строки, даже пустые (без данных).
- **-b** - включить нумерацию непустых строк (заполненных данными).
- **-E** -в конце каждой строки будет отображаться символ $.
- **-s** - выводить не более одной пустой строки при повторе.

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран содержимое файла file1
   a. **[root@host -]# cat file1**

2. Соединить данные из file2 с file1 в файле file3
   a. **[root@host -]# cat file1 file2 > file3**

3. Вывести на экран все непустые строки файла /etc/profile
   a. **[root@host -]# cat -b /etc/profile**

4. Вывести на экран содержимое файла /etc/profile с номерами строк
   a. **[root@host -]# cat -n /etc/profile**

5. Вывести на экран содержимое файла /etc/profile удалив повторяющиеся пустые строки
   a. **[root@host -]# cat -s /etc/profile**

more

Команда more предназначена для постраничного просмотра файлов. Своим названием она обязана надписи more (в русскоязычном варианте — дальше), появляющейся внизу каждой страницы.

Команда имеет следующий синтаксис

more [опции] [имя_файла]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-n** - отображение n-го количества строк;
- **+n** - отображение текста, начиная со строки с номером n;
- **-c** -устранение потребности в прокрутке — отображение текста, начиная с верха экрана, и стирание при этом предыдущего вывода построчно;
- **-s** - замена нескольких пустых строк, расположенных подряд, одной пустой строкой;

В таблице 9 представлены некоторые полезные горячие клавиши для работы.

Таблица 9. Ключевые команды для работы с more

Горячая клавиша	Эффект
Enter	Спустится на одну строку вниз
Ctrl+F	Опустить текст на один экран
Пробел	Опустить текст на один экран
Ctrl+B	Поднять текст на один экран
b	Поднять текст на один экран
=	Вывести текущий номер строки
:f	Вывести имя файла и текущий номер строки
q	Выйти из more

less

Команда схожая с more, она позволяет перематывать текст не только вперёд, но и назад, осуществлять поиск в обоих направлениях, переходить сразу в конец или в начало файла. Особенность less заключается в том, что команда не считывает текст полностью, а загружает его небольшими фрагментами.

Команда имеет следующий синтаксис **less [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-f** - открыть специальный файл;
- **-i** - игнорировать регистр;
- **-s** - заменить множество идущих подряд пустых строк одной пустой строкой;
- **-N** - вывести номера строк;
- **-o** - сохранить выведенный контент в файл.

В таблице 10 представлены некоторые полезные горячие клавиши для работы.

Таблица 10. Ключевые команды для работы с less

Горячая клавиша	Эффект
b	Перейти на экран вверх (Page up)
d	Перейти на экран вниз
h	Показать справочные данные по работе с приложением
q	Выйти из приложения
Стрелки вверх/вниз	Подняться/опуститься на одну линию

4.3.2 ИЗВЛЕЧЕНИЕ ТЕКСТОВЫХ ДАННЫХ

head

Команда head выводит начальные строки (по умолчанию — 10) из одного или нескольких документов. Также она может показывать данные, которые передает на вывод другая утилита.

Команда имеет следующий синтаксис

head [опции] [имя_файла]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-q** - выводит только текст, не добавляя к нему название файла
- **-v** - перед текстом выводит название файла
- **-c** - позволяет задавать количество текста не в строках, а в байтах
- **-n** - показывает заданное количество строк вместо 10, которые выводятся по умолчанию.

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран первый 20 строк файла /etc/passwd
   a. **[root@host -]# head –n 20 /etc/passwd**

2. Вывести на экран все содержимое файла /etc/passwd кроме 20 первых последних строк
   a. **[root@host -]# head –n -20 /etc/passwd**

tail

Команда tail позволяет выводить заданное количество строк с конца файла, а также выводить новые строки в интерактивном режиме
Команда имеет следующий синтаксис **tail [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-q** - не выводить имена файлов
- **-f** - обновлять информацию по мере появления новых строк в файле
- **-v** - выводить подробную информацию о файле
- **-c** - выводить указанное количество байт с конца файла
- **-n** - выводить указанное количество строк из конца файла

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран последние 20 строк файла /etc/passwd
   a. **[root@host -]# tail –n 20 /etc/passwd**

2. Вывести вы данные кроме первых 20 строк
   a. **[root@host -]# tail –n +20 /etc/passwd**

cut

Команда cut используется, если нужно вырезать часть текста — при этом он может находиться в файле либо быть напечатанным через стандартный ввод. Команда удаляет секции текста, которые были обозначены при помощи байтов, символов или полей, разделенных знаками "-" и ":".
Команда имеет следующий синтаксис **cut [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-b** - номер байта, набор или диапазон байтов, подлежащих вырезанию;
- **-c** - символ, который следует вырезать. Также можно указывать набор либо диапазон символов;
- **-d** - с помощью этой опции пользователь устанавливает свой разделитель вместо стандартного TAB;
- **-f** - перечень полей для вырезания.

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран первую колонку файла /etc/passwd. В качестве разделителя использовать ":"
   a. **[root@host -]# cut -d: -f1 /etc/passwd**

2. Вывести на экран только с 3 по 8 символы каждой строки файла /etc/passwd
   a. **[root@host -]# cut -c 3-8 /etc/passwd**

awk

Команда awk - один из самых мощных инструментов для обработки и фильтрации текста, доступный даже для людей никак не связанных с программированием. Это не просто утилита, а целый язык разработанный для обработки и извлечения данных.

Команда имеет следующий синтаксис **awk [действие] [имя_файла]**

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран пять аккаунтов которые недавно подключались к системе
   a. **[root@host ~]# last -n 5 | awk '{print $1}'**

grep

Команда grep предназначена для поиска строк, соответствующих шаблону, заданному в параметре. Каждая найденная строка записывается в стандартный поток вывода.

Команда имеет следующий синтаксис **grep [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-c** - вывести количество найденных строк;
- **-i** - не учитывать регистр символов;
- **-w** - искать шаблон как слово, отделенное пробелами или другими знаками препинания;
- **-x** - искать шаблон как целую строку, от начала и до символа перевода строки.

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран первую строку содержащую openscaler в /etc/passwd.
   a. **[root@host -]# cat /etc/passwd | grep openscaler**

wc

Команда позволяет подсчитать количество строк или слов в тексте.

Команда имеет следующий синтаксис **wc [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-c** - отобразить размер файла в байтах
- **-l** - вывести количество строк в файле
- **-w** - отобразить количество слов в файле

Примеры наиболее распространенных случаев использования команды:

1. Вывести на экран количество слов, строк в файле /etc/passwd
   a. **[root@host -]# cat /etc/passwd | wc -wlc**

sort

Команда sort используется для вывода текстовых строк в определенном порядке, т.е. их сортировки. Ее можно использовать для сортировки текста из одного или нескольких файлов или с помощью нее может быть выполнена сортировка вывода для какой-либо другой команды.

Команда имеет следующий синтаксис **sort [опции] [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-b** - не учитывать пробелы;
- **-c** - проверить был ли отсортирован файл;
- **-d** - использовать для сортировки только буквы и цифры;
- **-n** - сортировка строк по числовому значению;
- **-r** - сортировать в обратном порядке;
- **-o** - вывести результат в файл;
- **-u** - игнорировать повторяющиеся строки.

diff

Команда diff служит для сравнения двух или нескольких файлов. Она может сравнивать как отдельные файлы, так и каталоги.
Команда имеет следующий синтаксис

diff [опции] [имя_файла_или_директории_1] [имя_файла_или_директории_2]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-b** - игнорировать изменения, связанные с добавлением пробелов;
- **-B** - игнорировать новые пустые строки;
- **-c** - вывести все данные и отметить изменения;
- **-r** - просматривать каталоги рекурсивно;
- **-w** - игнорировать изменения, связанные с добавлением пробелов и табуляции.

Примеры наиболее распространенных случаев использования команды:

1. Сравнить файлы f1 и f2 и вывести на экран только различия
   a. **[root@host -]# diff f1 f2**

2. Сравнить файлы f1 и f2 и вывести на экран все данные отметив различия
   a. **[root@host -]# diff -c f1 f2**

3. Сравнить все файлы в директориях d1 и d2
   a. **[root@host -]# diff -r d1 d2**

tr

Команда tr используется для замены, замещения или удаления символов из стандартного ввода, отправляя результат на стандартный вывод.

Команда имеет следующий синтаксис **tr [опции] [имя_набора_1] [имя_набора_2]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-c** - Сначала получить дополнение набора1;
- **-d** - Удалить знаки из набора2;
- **-s** - Замещать последовательность знаков, которые повторяются, из перечисленных в последнем наборе;
- **-t** - Сначала сократить набор1 до размеров набора2.

Примеры наиболее распространенных случаев использования команды:

1. Заменить строчные буквы на прописные в файле text.txt
   a. **[root@host -]# cat text.txt | tr a-z A-Z**

sed

Команда sed - это потоковый редактор текста, работающий по принципу замены. Его можно использовать для поиска, вставки, замены и удаления фрагментов в файле. С помощью этой утилиты возможно редактировать файлы не открывая их.

Команда имеет следующий синтаксис **sed [опции] {действие} [имя_файла]**

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-n** - не выводить содержимое буфера шаблона в конце каждой итерации;
- **-e** - команды, которые надо выполнить для редактирования;
- **-f** - прочитать команды редактирования из файла;
- **-i** - сделать резервную копию файла перед редактированием;
- **-l** - указать свою длину строки;
- **-r** - включить поддержку расширенного синтаксиса регулярных выражений.

Примеры наиболее распространенных случаев использования команды:

1. Вывести данные файла /etc/passwd за исключением строк 2 - 5.
   a. **[root@host ~]# cat /etc/passwd | sed '2,5d'**
      i. **Здесь:**
         1. **"2,5d" - означает удалить (d) строки 2-5**

2. Вывести данные файла /etc/passwd - только первые 10 строк
   a. **[root@host ~]# cat /etc/passwd | sed '11,$d'**
      i. **Здесь:**
         1. **"11,$d" - означает удалить строки с 11 до конца документа**

3. В файле /etc/passwd после второй строки вставить строку “openscaler”
   a. **[root@host ~]# cat /etc/passwd | sed '2a openscaler'**
      i. **Здесь:**
         1. **"a" - означает добавить**

4. Вывести на экран содержимое файла /etc/passwd с 10 по 20 строки
   a. **[root@host ~]# cat /etc/passwd | sed -n '10,20p'**

5. Заменить содержимое строк с 10 по 20 в файле /etc/passwd на “openscaler”
   a. **[root@host ~]# cat /etc/passwd | sed '10,20c openscaler'**

4.4 ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

- Какова разница между командным режимом и режимом ввода в редакторе vi?
- Как переключаться между режимами работы редактора vi?
- В чем разница и схожести в командах cut и grep?
- Каким образом можно заменить значение определенных строк в файле?

5. УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И РЕГЛАМЕНТИРОВАНИЕ ПРАВ ДОСТУПА

В Linux для каждого пользователя фиксируется как минимум следующий объем информации - имя пользователя, пароль, домашняя директория. В системе существует несколько типов пользователей, основной администратор системы (пользователь root), специальные пользователи создаваемые самой систем (для определенных служебных нужд) и обычные пользователи системы. Для упрощения процесса управления правами доступа и регламентирования действий пользователей в системе используется концепция “групп пользователей”, в которую включаются пользователи с одинаковыми правами на определенные действия или уровнем доступа. Каждый из системных пользователей принадлежит как минимум одной группе.

Правильное управление правами пользователей в системе и назначением групп является важной частью обеспечения комплексной безопасности операционной системы. В данной главе будут рассмотрены основные принципы управления пользователями и их группами с целью регламентирования их прав доступа в дистрибутиве OpenScaler Linux.

5.1 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И ГРУППАМИ

5.1.1 Управление пользователями

Под управлением пользователями в системе подразумевается выполнение следующих действий: их создание, удаление, модификация учетных и сопроводительных данных.

Для создания нового пользователя в системе должны быть выделены его ID, основная группа, домашняя директория и тип оболочки по умолчанию.

5.1.1.1 Добавление пользователя

Для добавления пользователя в систему используется команда useradd доступная суперпользователю root.

Команда имеет следующий синтаксис **useradd [опции] [имя_пользователя]**

К примеру, для создания пользователя “openscaler” необходимо выполнить следующую команду:

[root@host -] # useradd openscaler

Если после выполнения команды никаких дополнительных сообщений не было отображено, команда выполнилась успешно и пользователь был создан. Следующим шагом является задание пароля для вновь созданного пользователя для этого выполните команду

```
[root@host -] # passwd openscaler
Changing password for user openEuler.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Если при создании пароля выдается сообщение

"BADPASSWORD: The password fails the dictionary check”

значит избранный пароль создан на базе общеупотребимого слова или слишком простой. Попробуйте его усложнить, добавив специальные символы.

В OpenScaler можно выполнить команду

cracklib-unpacker /usr/share/cracklib/pw_dict > dictionary.txt,

чтобы экспортировать файл библиотеки словарей dictionary.txt. Вы можете проверить, находится ли пароль в этом словаре.

При задании пароля необходимо помнить о базовых требованиях к нему со стороны операционной системы:

- Пароль должен содержать как минимум восемь символов
- Пароль должен содержать как минимум 3 типа символов: строчные и прописные буквы, цифры, специальные символы
- Пароль не должен совпадать с именем пользователя
- Пароль не должен состоять из общеупотребимых слов

Выполните команду id, чтобы просмотреть информацию о вновь созданном пользователе.

```
[root@host -]# id openscaler
uid=1000(openscaler) gid=1000(openscaler) groups=1000(openscaler)
```

5.1.1.2 Изменение учетной записи пользователя

Под изменением учетной записи пользователя понимается следующий возможный набор действий: смена пароля, стандартной оболочки, домашней директории, смена UID и установка периода активности данной учетной записи.

Смена пароля

Обычные пользователи системы могут самостоятельно менять свой собственный пароль к учетной записи, используя команду **passwd**.

Суперпользователь (root) имеет возможность в принудительном порядке менять пароли для любого пользователя системы.

Смена оболочки

Обычные пользователи системы могут самостоятельно менять используемую ими оболочку для своей учетной записи, используя команду **chsh.**

Суперпользователь (root) имеет возможность в принудительном порядке менять оболочку, используемую для любого пользователя системы.

Также можно осуществить смену используемой оболочки с помощью команды usermod. Синтаксис команды следующий:

usermod -s путь_к_новой оболочке имя_пользователя

К примеру, смена оболочки на CSH для пользователя openscaler будет выглядеть следующим образом:

[root@host -]# usermod -s /bin/csh openscaler

Смена домашней директории пользователя

Смена домашней директории для пользователей системы доступна только для суперпользователя (root) и осуществляется с помощью следующей команды:

usermod -d путь_к_домашней_директории имя_пользователя

Если при смене домашней директории пользователя также планируется переместить все ее текущее содержимое в новую директорию, необходимо использовать дополнительно опцию **-m** как представлено в примере ниже.

usermod -d путь_к_домашней_директории -m имя_пользователя

Смена UID пользователя

Для того чтобы сменить ID пользователя необходимо использовать следующую команду **usermod -u UID имя_пользователя**

Все файлы и директории которыми обладал пользователь на момент смены UID также изменят свою принадлежность, поменяв исходный UID на новый. Но это работает только для тех файлов которые размещены в его домашнем каталоге. Для файлов принадлежащих пользователю и расположенных вне домашнего каталога - права доступа придется изменять самостоятельно используя команду **chown**.

Смена периода активности учетных данных пользователя

Данную манипуляцию может осуществлять исключительно суперпользователь (root). Для того, чтобы задать период активности учетных данных пользователя (после которого учетные данные будут заблокированы и возможности авторизоваться с ними более не будет), необходимо выполнить следующую команду:

usermod -e MM/DD/YY имя_пользователя

Где:

- MM - месяц
- DD - день
- YY - год

5.1.1.3 Удаление учетной записи пользователя

Для процедуры удаления учетных данных пользователя из системы потребуется суперпользователь (root). Используется для этого команда userdel.

К примеру, удаление пользователя Test будет выглядеть следующим образом:

[root@host -]# userdel Test

Если вместе с удалением учетных данных пользователя также предполагается удалить его домашнюю директорию и все ее содержимое, необходимо добавить к команде опцию **-r**.

Обычно удаление пользователей проводится в тот момент, когда они не авторизованы в системе. Если необходимо удалить учетные данные пользователя, не дожидаясь его выхода из системы, добавьте к команде опцию **-f**.

5.1.1.4 Использование административных функций обычными пользователями системы

Использование команды **sudo** позволяет обычным пользователям использовать команды, ориентированные на использование администраторами системы и суперпользователем (root).

Соответствующие разрешающие права для каждого из пользователей должны быть прописаны в файле **/etc/sudoers**. К примеру, будучи прописанным в файле sudoers пользователь, используя команду sudo, может осуществить создание нового пользователя, как показано на примере ниже.

[root@host -]#sudo /usr/sbin/useradd newuserl

Также в файле **/etc/sudoers** прописываются и ограничения, какие административные команды может использовать пользователь, а какие нет.

Примерный вид конфигурационного файла /etc/sudoers представлен ниже

```
[root@host ~]# cat /etc/sudoers
Defaults !visiblepw
Defaults env_reset
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS"
Defaults env_keep += "MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE"
Defaults env_keep += "LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES"
Defaults env_keep += "LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER
LC_TELEPHONE"
Defaults env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
XAUTHORITY"
Defaults secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
root ALL=(ALL) ALL
## Allows people in group wheel to run all commands
%wheel ALL=(ALL)
 ALL
```

Обратите внимание на строки, отмеченные “**##**” - данные строки являются комментариями и не имеют функционального назначения при чтении конфигурационного файла системой.

5.1.1.5 Основные конфигурационные файлы для учетных записей пользователей

Ключевыми конфигурационными файлами важными для работы с пользователями в системе являются:

- **/etc/passwd** - содержит данные пользователей
- **/etc/shadow** - файл, в котором хранятся фактические пароли пользователей в зашифрованном формате
- **/etc/default/useradd** - перечень настроек по умолчанию применяемых командой useradd при создании новых пользователей.
- **/etc/skel** - используется для запуска домашнего каталога при создании пользователя

5.1.2 Управление группами пользователей

В системе OpenScaler каждый пользователь является членом как минимум одной группы со своими правами в рамках системы. Суперпользователь (root), используя назначение пользователей в те или иные группы, может централизованно регламентировать действия типовых пользователей в системе.
Под управлением группами пользователей подразумевается их создание, изменение и удаление.

Создание новой группы пользователей

Суперпользователь (root) может создать новую группу пользователей, используя команду **groupadd**, чей синтаксис представлен ниже.

groupadd [опции] имя_группы

К примеру, создание новой группы пользователей под названием “usergroupexample” выглядит следующим образом - **[root@host -] # groupadd groupexample**

Изменение ID группы (GID)

Суперпользователь (root) может изменить GID группы пользователей, используя команду **groupmod** чей синтаксис представлен ниже.

groupmod -g GID имя_группы

Изменение имени группы пользователей

Суперпользователь (root) может изменить имя группы пользователей, используя команду **groupmod** чей синтаксис представлен ниже.

[root@host -] groupmod -n новое_имя_группы старое_имя_группы

Удаление группы пользователей

Суперпользователь (root) может удалить группу пользователей, используя команду **groupdel** чей пример использования представлен ниже.

[root@host -] # groupdel Test - удаление группы Test

Добавление/удаление пользователя из группы пользователей

Суперпользователь (root) может добавить/удалить пользователей в группу пользователей, используя команду **gpasswd**, чей пример использования представлен ниже.

[root@host -]# gpasswd -a openscaler Test - добавление пользователя openscaler в группу Test
[root@host -]# gpasswd -d openscaler Test - удаление пользователя openscaler из группы Test

Изменение основной группы пользователя

Если пользователь одновременно принадлежит к нескольким группам, его основную группу можно выбрать из состава тех в которых он состоит. Для этого можно использовать команду **newgrp**

К примеру, следующая команда выбирает в качестве основной группы группу Test для пользователя openscaler

```
[root@host -]# newgrp Test
Welcome to 4.19.90-2003.4.0.0036.oe1.x86_64
System information as of time: Mon Dec 14 14:36:45 CST 2020
System load: 0.00
Processes: 85
Memory used: 4.5%
Swap used: 0.0%
Usage On: 7%
IP address: 192.168.0.90
Users online: 1
```

Для настройки групп пользователей ключевыми являются следующие конфигурационные файлы:

- **/etc/gshadow** - информация о группах в шифрованном виде
- **/etc/group** - файл описания групп
- **/etc/login.defs** - основные системные настройки пользовательского профиля

5.2 УПРАВЛЕНИЕ ПРАВАМИ ДОСТУПА К ФАЙЛАМ

5.2.1 Базовые механизмы управления правами доступа

В операционной системе openscaler вы можете выполнить команды **ll** или **ls -l** для того чтобы ознакомится с атрибутами интересующего вас файла или каталога.

```
[root@host bin]# ls -l
total 97224
-rwxr-xr-x. 1 root root 55624 Mar 24 2020 '['
-rwxr-xr-x. 1 root root 39248 Mar 24 2020 addftinfo
-rwxr-xr-x. 1 root root 35488 Mar 24 2020 addr2line
```

Как видно из примера выше, первый атрибут для файла addftinfo это "**-**" . Он означает, что это обычный файл. Но бывают и другие атрибуты, идентифицирующие тип файла. Они представлены в таблице 11.

Таблица 11. Атрибуты, определяющие типы файла

Атрибут	Тип файла
-	Обычный файл, не подпадающий под ниже представленные категории
D	Директория
B	Блочное устройство
C	Символьное устройство
L	Символическая ссылка
P	Именованный канал (Pipe)
S	Файл сокета (Soket)

Следующая группа из трех атрибутов означает следующее:

- **r** - доступно для чтения
  - Для файла - это означает возможность прочитать его содержимое
  - Для директории - это означает возможность увидеть ее поддиректории и файлы, в ней находящиеся

- **w** - доступно для записи
  - Для файла - возможность его редактирования, изменения и удаления
  - Для директории - возможность редактирования, изменения и удаления всех файлов и поддиректорий в ее составе

- **x** - доступно для исполнения
  - Для файлов - возможность их исполнения
  - Для директории - возможность доступа к ней

Зачастую символы RWX представляют числовыми аналогами. R равно 4, W равно 2, X равно 1. К примеру, для владельца файла права равны 7 (R+W+X), для пользователей, состоящих в одной группе с хозяином файла - 5 (R+X), для всех остальных- 5 (R+X). Тогда полный набор прав на данный файл или директорию представляется как 755.

5.2.2 Права доступа к файлам и директориям

Права на файлы и директории в системе разделены на три уровня - владелец, группа пользователей в которой состоит владелец и все остальные. Для управления правами доступа для всех трех уровней используется команда **chmod**

Задавать права доступа можно двумя путями - с использованием вышеописанных атрибутов RWX или их численными значениями.

Численные значения прав считаются следующим образом: к примеру, в системе присутствует файл с правами -rwxrwx-- из этого следует:

- Права для хозяина файла - rwx = 4 + 2 + 1 = 7
- Права для группы хозяина файла rwx = 4 + 2 + 1 = 7
- Права для всех остальных пользователей системы --- = 0 + 0 + 0 = 0

Где чтение (r) - 4, запись/изменение (w) - 2, исполнение (x) - 1

Таким образом численное значение прав доступа к данному файлу - 770.

Для задания прав доступа к файлу или каталогу можно использовать команду chmod, обладающую следующим синтаксисом:

chmod [-R] xyz имя_файла_или_директории

- Где **xyz** - численное значение прав доступа (в нашем случае 770)
- **-R** - означает что указанные права будут установлены на директорию и все файлы и поддиректории в ее составе.

К примеру, если вы хотите предоставить полный доступ для всех пользователей системы к файлу .bashrc, выполните следующую команду:

```
[root@host -]# ls -al .bashrc
-rw-r--r--. 1 root root 176 Oct 29 2019 .bashrc
[root@host -]# chmod 777 .bashrc
[root@host -]# ls -al .bashrc
-rwxrwxrwx. 1 root root 176 Oct 29 2019 .bashrc
```

Хотите изменить права на -rwxr-xr--? Тогда численное значение будет таковым: [4+2+1][4+0+1][4+0+0]=754.

Изменение прав доступа с помощью символьных атрибутов

Как мы уже ранее говорили, в системе есть три уровня прав - хозяин файла (**u**) или директории, группа хозяина (**g**) и все остальные пользователи системы (**o**). Для каждой из групп есть свой набор прав (чтение, запись, исполнение).

Используя символы **u g o** - мы можем передать chmod указание права, для какого из уровней мы редактируем.

Указав уровень согласно выше представленным символам, мы можем также и задать права используя r w x.

Ознакомьтесь с таблицей 12 с перечнем параметров, необходимых для использования символьного задания прав через chmod

Таблица 12. Символьное задание прав через chmod

Команда	Уровень прав	Действие	Разрешения	Объект
chmod	u (хозяин)	+ (добавить)	r (чтение)	Файл или каталог
	g (группа)	- (отнять)	w (запись)	
	o (others)	= (установить)	x (исполнение)	
	a (все)			

К примеру, чтобы установить права на файл command равные -rwxr-xr--, выполните команду **chmod u=rwx,g=rx,o=r command**

```
[root@host -]# touch test1
[root@host -]# ls -al test1
-rw------- 1 root root 0 Dec 14 14:43 test1
[root@host -]# chmod u=rwx,g=rx,o=r test1
[root@host -]# ls -al test1
-rwxr-xr-- 1 root root 0 Dec 14 14:43 test1
```

Хотите добавить или удалить права сразу для всех уровней доступа? Воспользуйтесь опцией **a**. К примеру, чтобы удалить права на исполнение файла сразу для всех трех уровней, используйте команду:

```
[root@host -]# chmod a-x test1
[root@host -]# ls -al test1
-rw-r--r-- 1 root root 0 Dec 14 14:43 test1
```

chown

Данная команда используется для изменения хозяина и группы хозяина файла и доступна исключительно суперпользователю (root).

Базовый синтаксис команды следующий:
chown [–R] имя_хозяина имя_файла
chown [-R] имя_хозяина:группа_хозяина имя_файла

Примеры наиболее распространенных случаев использования команды:

1. Перейти в корневую директорию (/), создать файл test.txt и назначить владельцем пользователя openscaler

```
[root@host /]# cd /
[root@host /]# touch test.txt
[root@host /]# chown openscaler test.txt
[root@host /]# ls -l
total 64
-rw------- 1 openscaler root 0 Dec 14 14:47 test.txt
```

2. Сменить владельца и группу файла test.txt обратно на root

```
[root@host /]# chown root:root test.txt
[root@host /]# ls -l
total 64
dr-xr-xr-x 13 root root 0 Dec 14 14:47 sys
-rw------- 1 root root 0 Dec 14 14:49 test.txt
```

chgrp

Данная команда используется для смены группы владельца файла и доступна только суперпользователю (root).
Команда обладает следующим синтаксисом:

chgrp [-cfhRv][--help][--version] [Группа] [Файл или директория]

Где:

- **-c** - при обработке выводить только изменения
- **-f** - не выводить сообщения об ошибках
- **-h** - работать непосредственно с самими символьными ссылками, а не с файлами, на которые они ссылаются
- **-R** - рекурсивная обработка каталога со всем его содержимым
- **-v** - выводить информацию о каждом обработанном файле
- **--help** - справка по использованию команды
- **--version** - справка по текущей версии программы

Примеры наиболее распространенных случаев использования команды:

1. Изменить группу владельца у файла

```
[root@host /]# chgrp -v openscaler test.txt
changed group of'test.txt' from root to openscaler
[root@host /]# ll
total 64K
-rw------- 1 root openscaler 0 Dec 14 14:49 test.txt
```

umask

Команда позволяет задавать права доступа по умолчанию на момент создания файла или директории. Команда доступна также и обычным пользователям системы.

Обладает следующим базовым синтаксисом:

umask [-S] [маска]

Где -S означает, что маска задается буквенными символами.

Чтобы увидеть текущую используемую маску, выполним команду:

```
[root@host /]# umask
0077
```

Создадим директорию и посмотрим стандартные права, ей присвоенные на момент создания.

```
[root@host /]# umask
0077
[root@host /]# mkdir test1
[root@host /]# ls -l -d /test1
drwx------ 2 root root 4096 Dec 14 14:53 /test1
```

Первая цифра маски ни на что не влияет и является пережитком синтаксиса языка C. Дальше цифры аналогичны правам доступа: первая - хозяин, вторая - группа и третья - все остальные. Эта маска используется для расчета прав файла. Если не вдаваться в подробности, то рассчитывается всё довольно просто, от максимальных прав отнимается маска и получаются права для файла. Фактически, получается, что маска содержит права, которые не будут установлены для файла. Поэтому права по умолчанию для файла"drwxr-xr-x"="777-077=700".

5.2.3 Специальные права доступа к файлам и директориям

В операционной системе также существует понятие “битов разрешений”, это **Setuid**, **Setgid** и **Sticky Bit**. Это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги.

Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо.

По умолчанию в системе присутствует всего один файл с заранее установленным setuid **- /etc/passwd**

```
[root@host /]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 31K Mar 24 2020 /usr/bin/passwd
```

Отмечаем символ “**s**” в правах файла.

К примеру, обычные пользователи не имеют возможности просмотреть содержимое домашней директории суперпользователя /root.

```
[openscaler@host -]$ ls /root
ls: Unable to open directory /root: Insufficient permissions
```

Применим Setid для утилиты ls

```
[root@host -]# which ls
alias ls='ls --color=auto'
/usr/bin/ls
[root@host -]# chmod u+s /usr/bin/ls
```

Это позволяет обычному пользователю системы выполнять команды с повышенными привилегиями без необходимости входа в систему как root

chmod u-s имя_файла- отменяет setuid

Setgid - Принцип работы Setgid очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.
Повторим предыдущий пример с доступом к каталогу /root на сей раз задав setgid (временно дать правда группе root).

```
[openscaler@host -]$ ls -l /root
ls: Unable to open directory /root: Insufficient permissions
[root@host -]# chmod g+s /usr/bin/ls
[root@host -]$ ls -l /root
```

Если setgid установлен на директорию, то все ее файлы и поддиректории создаваемые пользователями приобретают группу владельца идентичную родительской директории.

```
[root@host -]# mkdir abc
[root@host -]# chown :testx abc
[root@host -]# touch abc/a
[root@host -]# ls -l abc/
итого 0
-rw-r--r--. 1 root root 0 ноя 23 10:47 a
[root@host -]#chmod g+s abc
[root@host -]# mkdir abc/b
[root@host -]# touch abc/c
[root@host -]# ls -l abc
-rw-r--r--. 1 root root     0 ноя 23 10:49 a
drwxr-sr-x. 2 root testx 4096 ноя 23 10:51 b
-rw-r--r--. 1 root testx    0 ноя 23 10:51 c

[root@host -]# chmod g-s abc
[root@host -]# touch abc/d
[root@host -]# mkdir abc/e
[root@host -]# ls -l abc
итого 4
-rw-r--r--. 1 root root     0 ноя 23 10:49 a
drwxr-sr-x. 2 root testx 4096 ноя 23 10:51 b
-rw-r--r--. 1 root testx    0 ноя 23 10:51 c
-rw-r--r--. 1 root root     0 ноя 23 10:53 d
drwxr-xr-x. 2 root root  4096 ноя 23 10:53 e
```

Sticky Bit - В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. Пример использования этого бита в операционной системе - это системная папка /tmp . Эта папка разрешена на запись любому пользователю, но удалять файлы в ней могут только пользователи, являющиеся владельцами этих файлов.

```
[root@host /]# ls -ld /tmp
drwxrwxrwt 10 root root 240 Dec 14 14:28 /tmp
```

Все пользователи имеют установленные права RWT, предотвращая удаление файлов из директории всем, кроме их непосредственного хозяина.

Для его установки используется следующая команда

chmod o+t имя_файла/директории

5.2.4 Списки контроля доступа (ACL)

Одноуровневой модели прав доступа пользователей, которая применялась в Linux до появления ACL, на сегодняшний день явно недостаточно. Используя классическую модель прав доступа (хозяин, группа, все остальные), зачастую просто невозможно правильно распределить права доступа. Списки контроля доступа позволяют более гибко решить эту задачу. Девять бит плюс три специальных бита позволяют определить права доступа к файлу (чтение, запись, исполнение) только для трех классов пользователей – хозяин, группа и остальные. Такой механизм в большинстве случаев не пригоден для решения даже относительно простых задач. Чтобы определить доступ к какому-нибудь документу или ресурсу, пользователя обычно включают в определенную группу. Все, кто входит в эту группу, имеют одинаковые права, т.е. используется принцип “всё или ничего”. Списки контроля доступа ACL (Access Control Lists) позволяют установить права доступа к файлам не только для владельца и группы, но и индивидуально для любого другого пользователя или группы, без каких-либо ограничений по количеству устанавливаемых пользователей/групп.

В целом для создания и управления ACL достаточно будет освоить три команды **setfacl**, **getfacl**, и **chacl**.

getfacl – получить данные по текущему ACL для файла или каталога.
setfacl – основная команда для применения правил ACL к файлу или директории.

Формат применяемых правил выглядит следующим образом:

- **- [d[efault]:] [u[ser]:]uid [:perms]** – определяет разрешения для конкретного пользователя системы и хозяина файла
- **- [d[efault]:] g[roup]:gid [:perms]** - определяет разрешения для группы хозяина файла и всех остальных групп
- **- [d[efault]:] m[ask][:] [:perms]** - применяемая маска для разрешений пользователей по умолчанию
- **- [d[efault]:] o[ther] [:perms]** - определяет разрешения для остальных пользователей

Для формирования правила ACL необходимо задать имя пользователя или группы для которых данное правило направлено или их UID/GID. Поле “perms” включает в себя определение прав доступа (ранее рассмотренные нами права - RWX)

chacl - используется для изменения правил ACL для директории или файла.

Рассмотрим следующий пример:

```
[root@host -]# touch /test
[root@host -]# chmod 777 /test
[root@host ~]# getfacl /test // получить информацию о текущем ACL на файл
getfacl: Removing leading '/' from absolute path names
# file: test //File name
# owner: root //File owner
# group: root //Group to which the file belongs.
user::rwx //File owner permission
group::rwx //Rights of users in the same group
other::rwx //Other rights
```

Как вы можете увидеть, для всех остальных пользователей, не входящих в группу хозяина файла, установлены как RWX - чтение, запись и исполнение. Давайте изменим настройки ACL для данного файла применительно к пользователю code.

```
[root@host -]# setfacl -m u:code:r /test
[root@host -]# ll /test
-rwxrwxrwx+ 1 root root 1 Nov 22 09:10 /test
[root@host -]#
```

Снова проверим текущий ACL для данного файла

```
[root@host -]# getfacl /test
getfacl: Removing leading '/' from absolute path names
# file: test
# owner: root
# group: root
user::rwx
user:code:r--
group::rwx
mask::rwx
other::rwx
```

Заметим, что для пользователя code появилась своя строка с указанием прав - code:r--

Проверим доступ к файлу от имени пользователя code. Заметим сообщение -- INSERT -- W10: Warning: Changing a readonly file - так как пользователю на уровне ACL выданы только права на чтение.

Таким же образом можно ограничить права для пользователя, например только на изменение данных в файле. Сделаем это через изменение маски, оставив возможность только для записи.

```
[root@host ~]# setfacl -m m:w /test
```

Посмотрим как изменился состав правил ACL

```
[root@host -]# getfacl /test
getfacl: Removing leading '/' from absolute path names
# file: test
# owner: root
# group: root
user::rwx
user:code:r-- #effective:---
group::rwx #effective:-w-
mask::-w-
other::rwx
[root@host -]#
```

Внесем данные в файл test от имени пользователя root и попробуем получить к ним доступ под пользователем code

```
[root@host -]# echo "this is a test getfacl " >/test
[code@ localhost -]$ vim /test
"/test" [Permission Denied]
```

Отменим все ранее примененные изменения ACL

```
[root@host -]# setfacl -x u:code /test
[root@host -]# setfacl -x m /test
[root@host -]# getfacl /test
getfacl: Removing leading '/' from absolute path names
# file: test
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
[root@host -]# ll /test
The -rwxrwxrwx 1 root root 24 Nov 22 11:13 /test
```

5.2.5 Временная эскалация прав доступа

Как уже ранее обсуждалось в предыдущих главах, часть административных команд рассчитана исключительно только на выполнение от имени суперпользователя (root). Тем не менее, есть механизм получения временного доступа для исполнения этих команд обычными пользователями системы - команда **su**.

Данная команда заменяет пользователя оболочки shell на указанного. Фактически происходит запуск нового экземпляра оболочки с указанными параметрами.

Обычно команда используется в двух возможных вариантах вызова для смены учетной записи пользователя:

- **su** Если вызов команды происходит без аргументов, то происходит смена пользователя оболочки shell на суперпользователя root. Программа выдаст приглашение ввода пароля, если пароль будет верным, то текущим пользователем станет root
- **su -** В данном случае помимо смены пользователя происходит также смена контекста выполнения оболочки на контекст указанного пользователя. Переменные $PATH, $HOME, $SHELL, $USER, $LOGNAME содержат значения, характерные для указанного пользователя. Домашняя папка пользователя меняется на другую.

Команда обладает следующим синтаксисом:

su [-fmp] [-c команда] [-s оболочка] [--help] [--version] [-] [пользователь] [аргументы]

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-m -p** - не менять переменные окружения при выполнении команды
- **-c** - запускает приложение под указанным аккаунтом
- **-s** - происходит запуск для заданного пользователя указанной оболочки
- **--help** - описание команды и ее использования
- **--version** - вывод информации о версии
- **-l** - смена контекста выполнения на контекст заданного пользователя

sudo

Команда позволяет запускать программы от имени других пользователей, а также от имени суперпользователя (root).

Для ее использования пользователь должен быть указан в конфигурационном файле /etc/sudoers

Базовый синтаксис команды следующий:

sudo [опции] [-p prompt] [-u имя_пользователя/#uid] -s

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-V** - выводит версию утилиты
- **-h** - выполнить команду от имени другого хоста
- **-l** - позволяет вывести список доступных команд для удалённых пользователей
- **-k** - по умолчанию, sudo сохраняет пароль и некоторое время после выполнения команды, вы можете выполнить ещё раз, без ввода пароля. Эта опция отключает такую возможность
- **-b** - запускает переданную программу в фоновом режиме
- **-p** - использовать своё приглашение для ввода пароля
- **-u** - позволяет указать, от имени какого пользователя нужно выполнять программу
- **-s** - позволяет запустить указанный командный интерпретатор

Примеры наиболее распространенных случаев использования команды:

1. Запустить команду от суперпользователя (root)

   sudo ls
   [sudo] password for openScaler:
   openScaler is not in the sudoers file. This incident will be reported.

   Если пользователь внесен в конфигурационный файл /etc/sudoers то команда будет выполнена от имени суперпользователя (root)

2. Запустить команду от имени другого пользователя - userb
   a. **sudo -u userb ls -l**

3. Посмотреть текущие настройки sudo для пользователя

   $sudo -l //Display the sudo settings.
   Available options in a sudoers ``Defaults'' line:
   syslog: Syslog facility if syslog is being used for logging

4. Отредактировать файл www/index.html в домашнем каталоге пользователя от имени пользователя uggc
   a. **sudo -u uggc vi -www/index.html**

5. Посмотреть версию sudo
   a. **sudo –V**

5.3 Вопросы для самопроверки

- Создайте две группы пользователей с GID - 1010, 1020
- Создайте пользователя user1 и добавьте его в две новые созданные группы с GID 1010, 1020
- Создайте пользователя user2 назначте основной группой для него “mg”
- Установите время действия паролей для пользователей user1 и user2 в 30 календарных дней с уведомлением о необходимости смены пароля за 3 дня до прекращения его действия.
- Создайте файл test.txt в каталоге /tmp. Хозяин файла и группа должны быть - root. Дайте доступ на чтение данного файла пользователю user1 и на чтение и запись пользователю user2

6. УСТАНОВКА ДОПОЛНИТЕЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И УПРАВЛЕНИЕ СЛУЖБАМИ ОС

6.1 ОБЗОР СИСТЕМЫ УПРАВЛЕНИЯ ПРОГРАММНЫМИ ПАКЕТАМИ

Программные пакеты в Linux обычно подразделяются на две ключевые группы.

Бинарные пакеты представляющие собой скомпилированный и готовый к установке и использованию набор программного обеспечения и/или системных библиотек и пакеты с исходными текстам приложений и библиотек для их последующей компиляции пользователем самостоятельно. Таким образом с целью установки нового программного обеспечения возможно применение одного из двух подходов - установка бинарных пакетов содержащих уже скомпилированное и готовое к использованию приложение либо воспользовавшись исходными текстами провести компиляцию требуемого приложения самостоятельно.

В различных дистрибутивах могут использоваться разные форматы данных пакетов - RPM (Red Hat Enterprise Linux, CentOS, Alma Linux, openEuler, OpenScaler), DEB (Debian, Ubuntu), TGZ (Arch Linux) и многие другие.

В рамках данного курса мы будем рассматривать работу именно с программными пакетами формата RPM поскольку именно он используется в дистрибутивах OpenScaler и openEuler и является наиболее распространенным в среде корпоративных Linux дистрибутивов.

6.2 ОБЗОР ФОРМАТА БИНАРНЫХ ПАКЕТОВ RPM

RPM – это мощный инструмент для установки, обновления и удаления программного обеспечения в системе.

В качестве основных преимуществ использования бинарных установочных пакетов RPM стоит отметить следующее:

- Простота установки и последующего обновления программного обеспечения. С помощью RPM пакетов возможно легко устанавливать и обновлять программное обеспечение прямо из репозиториев (хранилищ программных пакетов) или с использованием командной строки.
- Однородность и совместимость: RPM обеспечивает стандартизацию и совместимость между различными дистрибутивами GNU/Linux. Это означает, что пакеты, созданные для одного дистрибутива, могут быть использованы на других дистрибутивах с поддержкой формата RPM без необходимости перекомпиляции исходного кода. В качестве примера можно отметить совместимость установочных бинарных пакетов дистрибутивов openEuler и OpenScaler, а также части пакетов собранных для дистрибутивов CentOS и семейства Red Hat
- Централизованное управление. С помощью утилиты управления пакетами, такой как **rpm** или **dnf**, пользователи могут легко управлять установленным программным обеспечением. Это включает в себя установку, обновление и удаление пакетов, а также управление зависимостями между пакетами.
- Безопасность. Установочные пакеты RPM подписываются цифровыми сертификатами для обеспечения целостности и аутентичности. Это позволяет пользователям проверять подлинность пакетов перед их установкой и обновлением, предотвращая возможную установку сомнительных пакетов.
- Система зависимостей. RPM формат поддерживает систему зависимостей, которая позволяет определить требования к другим пакетам для успешного запуска и исполнения программного обеспечения. Это гарантирует правильную установку и обновление всех необходимых зависимостей, что способствует стабильности и надежности системы.

Все пакеты, вне зависимости от типа (бинарны или с исходными текстами) имеют расширение *.rpm.

Имя пакета задается следующим образом:

"имя-программы"-"версия"-"релиз"."платформа или src".rpm

К примеру, пакет называется так: icewm-0.9.25-1.i386.rpm, т.е. icewm версии 0.9.25, релиз первый, для платформы PC-x86.

Присутствие в название аббревиатуры **src** (Source) означает что пакет содержит исходные коды приложения или библиотеки а не бинарную скомпилированную версию ПО.

6.2.1 Базовые параметры для управления пакетами RPM с помощью одноименной утилиты

Базовой командой - менеджером пакетов для основанных на RPM дистрибутивов является команда rpm.

Команда обладает следующим базовым синтаксисом:
rpm [опции] имя_приложения

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-i** - отмечает, что должна быть произведения установка указанного ПО или пакета
- **-h** - выводить статус-бар демонстрирующих ход работы приложения
- **-v** - показать подробную информацию о ходе установки
- **-U** - произвести обновление указанного ПО или пакета
- **-q** - запрос, получение информации о пакете, установлен ли он в системе
- **-a** - опрашивает все установленные в системе пакеты
- **-V** - запрос версии установленного пакета в системе
- **-c** - отобразить все конфигурационные файлы пакета
- **-p** - опрашивает или проверять указанный программный пакет
- **-vh** - отображать процесс установки пакета в ходе его выполнения
- **-qpl** - показать файлы поставляемые в указанном пакете
- **-qpi** - показать описание программного пакета
- **-qf** - найти к какому программному пакету принадлежит указанный файл
- **-Va** - проверить все установленные программные пакеты и найти “потерянные файлы”
- **-qa** - Найти пакет по имени

Примеры наиболее распространенных случаев использования команды:

1. Установка программного пакета
   [root@host]# rpm -hvi dejagnu-1.4.2-10.noarch.rpm
   Warning: dejagnu-1.4.2-10.noarch.rpm: V3 DSA Signature: NOKEY, key ID db42a60e
   Get ready...

2. Показать информацию о пакете
   [root@host]# rpm -qi dejagnu-1.4.2-10.noarch.rpm

6.3 ОБЗОР ПАКЕТНОГО МЕНЕДЖЕРА DNF ДЛЯ УПРАВЛЕНИЯ ПАКЕТАМИ

Во многих Linux дистрибутивах основанных на RPM пакетах основным менеджером пакетов является приложение YUM. С его использованием становится возможным осуществлять автоматическую загрузку программных пакетов из репозиториев доступных в сети (хранилища программных пакетов дистрибутива), упрощается процесс учета зависимостей программных пакетов (все недостающие библиотеки требуемые для работы приложения будут автоматически определены и также установлены в системе), за счет чего в целом повышается эффективность управления установкой и обновления программного обеспечения.

Тем не менее, у приложения YUM достаточно низкая производительность, высокое потребление ОЗУ и медленный процесс разрешения зависимостей пакета. В качестве его альтернативы не обладающей данным набором проблем был создан менеджер пакетов DNF Пакетный менеджер DNF является стандартом де-факто в дистрибутивах OpenScaler и openEuler, будучи полностью “YUM-совместимым”, данный пакетный менеджер в полном объеме реализует функционал своего предшественника, а также поддерживает открытый API и систему внешних подключаемых модулей.

По умолчанию доступ к DNF и в целом управление установленными в системе программными пакетами является прерогативой исключительно суперпользователя (root)

6.3.1 Конфигурационный файл пакетного менеджера DNF

Основной конфигурационный файл пакетного менеджера DNF расположен в **/etc/dnf/dnf.conf**. Сам конфигурационный файл традиционно разбивается на две основные секции:

- Секция "main" содержит основные ключевые настройки DNF.
- Секция "repository" содержит настройки источников программных пакетов доступных для установки в систему (хранилища программных пакетов) называемых далее в документе “репозиториями” ("repository"). Их может быть настроено произвольное количество в зависимости от требований.

Поскольку DNF, как мы уже ранее отмечали, совместим с пакетным менеджером YUM, он также проверяет содержимое конфигурационных файлов, расположенных в директории **/etc/yum.repos.d** на предмет наличия дополнительных доступных репозиториев программных пакетов прописанных в файлах данной директории.

Таким образом, доступно два варианты конфигурирования репозиториев программных пакетов в системе - прописать новый репозиторий в основном конфигурационном файле DNF в секции “repository” или создать файл с расширением *.repo в директории /etc/yum.repos.d с описанием репозитория. Менеджер пакетов проверит обе локации на предмет наличия настроек репозиториев.

Настройка основных параметров

Разберем содержимое секции “main” в конфигурационном файле /etc/dnf/dnf.conf Пример содержимого данной секции представлен ниже:

```
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
```

Также в секции main могут присутствовать следующие конфигурационные параметры, представленные в таблице 13.

Таблица 13. Основные настройки пакетного менеджера DNF

Параметр	Описание
cachedir	Директория для хранения кэша в виде скачанных пакетов и ведения базы данных пакетного менеджера
keepcache	Возможные значения 1 и 0. Опередеяет, будут ли храниться в кэше все скачанные программные пакеты и файлы заголовки (headers) или нет
debuglevel	Возможные значения от 0 до 10. По умолчанию значение 2. Определяет, какой уровни детализации будет отладочный вывод пакетного менеджера
clean_requirements_ on_remove	Возможные значения true/false. По умолчанию true. Определяет, будет ли пакетный менеджер удалять более ненужные пакеты-зависимости если таковые появятся после удаления пользователем определенного программного обеспечения
best	Возможные значения true/false. По умолчанию true. При установке программного пакета и наличии нескольких их версий, пакетный менеджер будет стараться устанавливать только последние, наиболее свежие версии.
obsoletes	Возможные значения 1 и 0. По умолчанию 1. Определяет, обновлять ли пакеты, помеченные как устаревшие (obsolete)
gpgcheck	Возможные значения 1 и 0. По умолчанию 1. Определяет, проводить ли GPG верификацию каждого скачанного пакета до его установки в систему
plugins	Возможные значения 1 и 0. По умолчанию 1. Определяет, включать ли поддержку сторонних расширений пакетного менеджера (plug-ins)
installonly_limit	Задает количество одновременно устанавливаемых пакетов. По умолчанию равен трем.

Настройка репозиториев программных пакетов

Как мы ранее проговорили, есть два пути задания дополнительных репозиториев программных пакетов. Прописать новый репозиторий в основном конфигурационном файле DNF в секции "**repository**" или создать файл с расширением ***.repo** в директории **/etc/yum.repos.d** с описанием репозитория. Менеджер пакетов проверит обе локации на предмет наличия настроек репозиториев.

Продемонстрируем, как настроить репозиторий программных пакетов, отредактировав основной конфигурационный файл пакетного менеджера DNF **/etc/dnf/dnf.conf**

Минимальный вариант описания репозитория представлен на примере ниже:

```
[repository]
name=repository_name
baseurl=repository_url
```

В качестве репозиториев программных пакетов можно использовать таковые дистрибутива OpenScale или OpenEuler, доступные по следующим ссылкам:
https://repo.openEuler.org/
https://repo.openscaler.ru/

По данным ссылкам надо выбрать необходимый репозиторий программных пакетов соответствующей установленной версии операционной системы и архитектуры процессора, на котором работает ОС.

В таблице 14 представлено описание данных параметров.

Таблица 14. Описание параметров описания репозитория программных пакетов

Параметр	Описание
name=repository_name	Текстовая строка с уникальным именем репозитория. Все репозитории обязаны иметь свое наименование, не конфликтующее с другими.

baseurl=repository_url	Адрес расположения репозитория программных пакетов
	• Если речь о сетевой репозитории, то должен быть
	  • http адрес - http://path/to/repo
	  • ftp адрес - ftp://path/to/repoLocal
	• Также может быть указано локальное расположение репозитория, если он размещен на данной машине или смонтирован в каталог
	  • file:///path/to/local/repo

После установки системы у вас уже есть как минимум один прописанный официальный репозиторий. Ознакомимся с содержимым его описания.

vi /etc/yum.repos.d/openScaler_aarch64.repo

(имя репозитория может быть другим, в случае использования к примеру версии для архитектуры X86_64)

```
[OS]
name=OS
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/OS/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/OS/$basearch/RPM-GPG-KEY-openScaler

[everything]
name=everything
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/everything/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/everything/$basearch/RPM-GPG-KEY-openScaler

[EPOL]
name=EPOL
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/EPOL/main/$basearch/
enabled=0
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/OS/$basearch/RPM-GPG-KEY-openScaler

[debuginfo]
name=debuginfo
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/debuginfo/$basearch/
enabled=0
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/debuginfo/$basearch/RPM-GPG-KEY-openScaler

[source]
name=source
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/source/
enabled=0
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/source/RPM-GPG-KEY-openScaler
```

```
[update]
name=update
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/update/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/OS/$basearch/RPM-GPG-KEY-openScaler

[update-source]
name=update-source
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/update/source/
enabled=0
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/source/RPM-GPG-KEY-openScaler
```

Опция “enabled” - определяет, использовать ли данный репозиторий для поиска и установки пакетов.

6.3.2 Отображение текущей конфигурации DNF

Чтобы посмотреть текущие настройки пакетного менеджера, выполните следующую команду

dnf config-manager --dump

Для того чтобы посмотреть список ID доступных репозиториев программных пакетов, выполните команду

dnf repolist

Выполните следующую команду, чтобы посмотреть информацию о репозитории программных пакетов, указав ранее определенных ID

dnf config-manager --dump repository

6.3.3 Управление установкой/удалением программных пакетов

В качестве ключевого менеджера пакетов в системе используется dnf именно на его примере и будут рассмотрены вопросы управления программными пакетами в системе.

6.3.3.1 Поиск программных пакетов

Используя dnf, пользователь может осуществлять поиск программного пакета по его имени или описанию. Команда используемая для осуществления поиска программных пакетов по доступным репозиториям представлена ниже.

dnf search <поисковый запрос>

К примеру, осуществим поиск пакета с веб-сервером, используя вышеназванную команду:

```
[root@host /] $dnf search httpd
========================================= N/S matched: httpd
=========================================
httpd.aarch64: Apache HTTP Server
httpd-devel.aarch64: Development interfaces for the Apache HTTP server
httpd-manual.noarch: Documentation for the Apache HTTP server
httpd-tools.aarch64: Tools for use with the Apache HTTP Server
libmicrohttpd.aarch64: Lightweight library for embedding a webserver in applications
mod_auth_mellon.aarch64: A SAML 2.0 authentication module for the Apache Httpd Server
mod_dav_svn.aarch64: Apache httpd module for Subversion server
```

6.3.3.2 Получение списка всех установленных пакетов

Для того чтобы получить информацию по всем установленным в системе программным пакетам, выполните команду:

dnf list installed

Для того чтобы вывести информацию о каком-либо отдельном пакете или группе пакетов, используйте команду ниже

dnf list <выражение для поиска>

К примеру:

```
[root@host /]$ dnf list httpd
Available Packages
httpd.aarch64 2.4.34-8.h5.oe1 Local
```

6.3.3.3 Получение информации о программном пакете

Для того чтобы получить справочную карточку по определенному программному пакету, необходимо выполнить следующую команду.

dnf info <имя_пакета>

К примеру,

Available Packages
Name: httpd
Version: 2.4.34
Release: 8.h5.oe1
Arch: aarch64
Size: 1.2 M
Repo: Local
Summary: Apache HTTP Server
URL: http://httpd.apache.org/
License: ASL 2.0
Description: The Apache HTTP Server is a powerful, efficient and extensible web server.

6.3.3.4 Установка программного пакета

Для того чтобы провести установку определенного программного пакета, необходимо выполнить следующую команду.

dnf install <имя_пакета>

К примеру,

[root@host /]# dnf install httpd

Если установка прерывается с ошибкой - ознакомьтесь с выводом команды, чтобы определить причину нештатной ситуации, кои могут быть - конфликт программных пакетов, конфликт файлов в составе пакета, неудовлетворенные зависимости пакетов и прочие.

6.3.3.5 Загрузка программных пакетов

Для того чтобы провести загрузку (но не установку) определенного программного пакета, необходимо выполнить следующую команду.

dnf download <имя_пакета>

Если необходимо загрузить не только сам программный пакет и все его зависимости, выполните команду, представленную ниже.

dnf download --resolve <имя_пакета>

К примеру,

[root@host /]# dnf download --resolve httpd

6.3.3.6 Удаление программных пакетов

Для того чтобы провести удаление (деинсталляцию) определенного программного пакета, необходимо выполнить следующую команду.

dnf remove <имя_пакета>

К примеру,

[root@host /]# dnf remove totem

6.3.3.7 Управление группами программных пакетов

Группой программных пакетов называется объединение программных пакетов по какому-либо признаку. Для упрощения процесса установки, можно провести установку всех пакетов, представляющих одну группу вместо индивидуальной (попакетной) установки.

Чтобы ознакомится какие есть группы пакетов доступные для установки и уже установленные в системе, необходимо выполнить команду

dnf groups summary

К примеру,

```
[root@host /]# dnf groups summary
dnf groups summary
Last metadata expiration check: 3:17:59 ago on Чт 23 ноя 2023 08:23:39.
Installed Groups: 1
Available Groups: 9
```

Для того чтобы вывести все существующие и доступные к установке группы, воспользуйтесь командой

```
dnf group list
```

К примеру,

```
[root@host /]# dnf groups list
Last metadata expiration check: 3:18:18 ago on Чт 23 ноя 2023 08:23:39.
Available Environment Groups:
        Сервер
        Узел виртуализации
Installed Environment Groups:
        Минимальная установка
Installed Groups:
        Разработка
        Available Groups:
        Контейнеры
        Рабочий стол GNOME
        Управление без монитора
        Программы совместимости с UNIX
        Сетевые сервера
        Поддержка научных утилит
        Утилиты безопасности
        Системные утилиты
        Поддержка Smart Card
```

Для того чтобы посмотреть какие подгруппы программные пакеты включает в себя группа, воспользуйтесь командой

```
command: dnf group info <имя_группы>
```

К примеру, посмотрим группу “Системные утилиты”

```
[root@host /]# dnf groups info "Системные утилиты"
Last metadata expiration check: 3:31:07 ago on Чт 23 ноя 2023 08:23:39.
Group: Системные утилиты
```

Description: Эта группа представляет собой набор различных инструментов для системы, таких как клиент для подключения к общим ресурсам SMB и инструменты для мониторинга сетевого трафика.
Default Packages:
- **chrony**
- **cifs-utils**
- **libreswan**
- **nmap**
- **openldap-clients**
- **samba-client**
- **setserial**
- **tigervnc**
- **tmux**
- **xdelta**
- **zsh**

Optional Packages:
- **PackageKit-command-not-found**
- **aide**
- **amanda-client**
- **arpwatch**
- **chrpath**
- **convmv**
- **createrepo_c**
- **environment-modules**
- **freerdp**
- **fuse**
- **gpm**
- **gssdp**
- **gupnp**
- **iotop**
- **lzop**
- **mc**
- **mtx**
- **net-snmp-utils**
- **oddjob**
- **oddjob-mkhomedir**
- **rear**
- **sysstat**
- **x3270-x11**

Установка группы пакетов

Как мы уже знаем, у каждой группы пакетов есть свое имя и ID. Вы можете указать их для команды установки группы пакетов, указанной ниже

dnf group install имя_или_ID_группы

К примеру, проведем установку группы пакетов “Development tools”

[root@host /]# dnf group install "Development Tools”

Удаление группы пакетов

Для удаления группы пакетов используйте команду, указанную ниже

dnf group remove имя_или_ID_группы

К примеру, проведем удаление группы пакетов “Development tools”

[root@host /]# dnf group remove "Development Tools”

Проверка и обновление пакетов

Менеджер пакетов DNF умеет опрашивать доступные ему репозитории программных пакетов на предмет наличия программных пакетов более новых версий готовых к установке. Он позволяет ознакомиться со списком доступных обновлений пакетов и, выбрав нужные, осуществить установку более свежих версий.

Для проверки доступных обновлений программных пакетов используйте команду, представленную ниже

dnf check-update

К примеру,

```
[root@host /]# dnf check-update
Last metadata expiration check: 3:41:28 ago on Чт 23 ноя 2023 08:23:39.
kubeadm.x86_64   1.28.2-0    kubernetes
kubectl.x86_64   1.28.2-0    kubernetes
kubelet.x86_64   1.28.2-0    kubernetes
```

Для того чтобы осуществить обновление одного из пакетов используйте команду, представленную ниже. (Доступна только для суперпользователя root)

dnf update <имя_пакета>

К примеру,

```
[root@host /]# dnf update kubeadm.x86_64
Last metadata expiration check: 3:43:14 ago on Чт 23 ноя 2023 08:23:39.
Dependencies resolved.
================================================================================
================================================================================
================================================================================
============================

 Package                 Architecture                 Version
 Repository                 Size
================================================================================
================================================================================
================================================================================
============================
 Upgrading:
  kubeadm                x86_64                       1.28.2-0
 kubernetes                 11 M

 Transaction Summary
================================================================================
================================================================================
================================================================================
============================
 Upgrade  1 Package

 Total download size: 11 M
 Is this ok [y/N]:
```

Для того чтобы обновить группу пакетов а не индивидуальный программный пакет, используйте команду, представленную ниже

```
dnf group update <имя_группы>
```

Если же требуется произвести обновление всей системы (всех установленных программных пакетов), используйте команду, представленную ниже (также доступна только для суперпользователя root)

```
dnf update
```

6.4 СБОРКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЗ ИСХОДНЫХ ТЕКСТОВ

Хотя в дистрибутивах OpenScaler и openEuler стандартом считается установка программного обеспечения используя бинарные пакеты RPM и менеджер пакетов DNF, иногда случаются ситуации, когда программное обеспечение должно быть скомпилировано из исходных кодов и установлено пользователем самостоятельно.
К примеру, это необходимо в следующих случаях:

- Версия программного обеспечение в RPM старше чем требуется а бинарный пакет с новой версией отсутствует
- Программное обеспечение в RPM пакете собрано с параметрами компиляции по каким-либо причинам, не устраивающим пользователя
- Программное обеспечение в принципе не имеет готовых RPM пакетов, к примеру некое самописное ПО пользователя системы
- Программное обеспечение в RPM пакете собрано с некоторым отключенным функционалом, требуемым пользователем
- Требуется пересборка ПО для оптимизации его производительности на нестандартном аппаратном обеспечении

Преимущества установки программного обеспечения из исходных кодов:

- В процессе компиляции при должном уровне знаний пользователь может максимально оптимизировать сборку с точки зрения производительности именно на его конкретном компьютере и требуемого функционала

Недостатки установки программного обеспечения из исходных кодов:

- Процесс настройки сборочной среды и компилятора должным образом достаточно сложный процесс, требующий знаний от пользователя
- Установка скомпилированного приложения из исходных текстов подразумевает самостоятельное отслеживание пользователем всех его библиотек-зависимостей
- Процесс обновления крайне затруднен и потребует ручного вмешательства пользователя. Скорее всего потребует повторения всех шагов с точки зрения самостоятельной сборки и установки новой версии, проверки и учета библиотек-зависимостей.

Таким образом пользователи сами вольны решать каким образом устанавливать программное обеспечение и какой подход применять. Тем не менее, очевидно что путь использования самостоятельной компиляции и установки ПО из исходных кодов должен применяться только в случае объективной необходимости, пользователем с достаточными техническими знаниями и точечно (только для определенного ПО, а не всех пакетов системы).

6.4.1 Установка программного обеспечения из исходных кодов

В самом простом и обобщенном случае процесс установки программного обеспечения из исходных кодов представляет собой следующую последовательность действий.

- Загрузите и распакуйте архив с набором исходных кодов требуемого приложения
- Ознакомьтесь с прилагаемым к исходным кодам текстовыми файлами README, INSTALLATION, SETUP. В них обычно указываются нюансы настройки процесса компиляции данного приложения, возможные опции для правильного конфигурирования и список зависимостей, требуемых для запуска собранного приложения
- Создайте “Makefile” выполнив скрипт **./configure**, указав требуемые опции конфигурирования согласно прочитанному в сопроводительных файлах
- Осуществите компиляцию бинарного исполняемого файла из исходных кодов, выполнив команду **make**
- Установите скомпилированное программное обеспечение, выполнив команду make install. При необходимости указав нужные параметры установки согласно прочитанному в сопроводительных файлах. По умолчанию каталог для установки приложения будет /usr/local/. Соответственно все ключевые конфигурационные файлы приложения по умолчанию будут располагаться в /usr/local/etc или /usr/local/XXX/etc.

С точки зрения выполняемых команд в самом тривиальном случае это выглядит следующим образом:

- Загрузим исходный код на наш компьютер в текущий каталог пользователя (текущий каталог пользователя можно узнать выполнив команду **pwd**)
  wget https://www.python.org/ftp/python/3.7.7/Python-3.7.7.tgz
- Распакуем загруженные исходные коды приложения, используя уже известный нам архиватор tar
  tar -zxvf Python-3.7.7.tgz
- Ознакомимся с прилагаемой сопроводительной документацией в файле README
  cat Python-3.7.7/README.rst
- Создадим “Makefile”, задав требуемые опции согласно прилагаемой документации. В данном примере мы объявляем, что собираемое ПО должно быть установлено не в путь по умолчанию, а в директорию/usr/local/Python
  ./configure --prefix=/usr/local/Python
- Скомпилируем программное обеспечени
  make
- Произведем установку скомпилированного программного обеспечения в систему
  make install

6.5 УПРАВЛЕНИЕ СЕРВИСАМИ ОПЕРАЦИОННОЙ СИСТЕМЫ (SYSTEMD)

В качестве стандартного менеджера сервисов операционной системы дистрибутив OpenScaler использует Systemd.

systemd — это менеджер системы и сервисов в операционной системе GNU/Linux. Он обеспечивает обратную совместимость со скриптами инициализации SysV и LSB и поддерживает такие функции, как активация сервисов на основе сокетов и D-Bus, активация демонов по запросу, моментальные снимки состояния системы и управление точками монтирования и автоматического монтирования. За счет systemd можно усовершенствовать логику управления и параллелизацию сервисов.

В systemd содержатся служебные модули, необходимые для управления всеми службами в составе ОС.

Компоненты классифицируются по представляемому ими типу ресурсов и определяются в файлах конфигурации компонентов. Например, компонент avahi.service представляет демон Avahi и определен в файле avahi.service.

Ключевые типы компонентов которыми оперирует менеджер сервисов представлены в таблице 15.

Таблица 15. Ключевые типы компонентов менеджера сервисов

Тип компонента	Расширение файла	Описание
Компонент сервиса	*.service	Системный сервис
Целевой компонент	*.target	Группа компонентов systemd
Компонент автоматического монтирования	*.automount	Точка автоматического монтирования файловой системы
Компонент устройства	*.device	Файл устройства, распознаваемого ядром
Компонент монтирования	*.mount	Точка монтирования файловой системы
Компонент пути	*.path	Файл или каталог в файловой системе
Компонент области	*.scope	Внешне создаваемый процесс

Тип компонента	Расширение файла	Описание
Компонент среза	*.slice	Группа иерархически упорядоченных компонентов, управляющих системными процессами
Компонент сокета	*.socket	Сокет межпроцессного взаимодействия
Компонент подкачки	*.swap	Устройство или файл подкачки
Компонент таймера	*.timer	Таймер systemd

Все доступные типы компонентов systemd по умолчанию размещены в одной из директорий описанных в таблице 16.

Таблица 16. Расположение ключевых компонентов менеджера сервисов systemd

Каталог	Описание
/usr/lib/systemd/system/	Компоненты systemd, распространяемые с установленными пакетами RPM.
/run/systemd/system/	Компоненты systemd, создаваемые во время выполнения.
/etc/systemd/system/	Компоненты systemd, создаваемые и контролируемые системным администратором.

Для управления менеджером сервисов systemd используется команда **systemctl**. Она позволяет в частности производить запуск, остановку, перезапуск, просмотр системных сервисов.

SystemD является более эффективным менеджером сервисом операционной системы нежели его предшественник - SysVinit.

Во многом это обусловлено следующими ключевыми преимуществами Systemd:

- **Параллельная активация нескольких сервисов системы**. Параллелизация в принципе отсутствует в SystemVinit, а в отличие от UpStart в SystemD она более агрессивная. Активация на основе сокетов и D-Bus уменьшает время загрузки операционной системы. Чтобы ускорить загрузку системы, SystemD в отличие от других вышеупомянутых менеджеров сервисов ОС пытается сделать следующее:
  - активировать только необходимые для системы и пользователя в данный момент процессы;
  - провести активацию наибольшего числа сервисов в параллельном режиме.

  Но важно помнить, что загрузка системы включает множество отдельных сервисов, некоторые из которых могут быть зависимыми друг от друга. Например, демон сетевой файловой системы (NFS) может запустится только после активации сетевого подключения. В systemd возможен параллельный запуск многих зависимых заданий, но не всех из них. К примеру - NFS. Невозможно смонтировать NFS и активировать сеть одновременно. Прежде чем запускать сервис, systemd вычисляет его зависимости, создает временную транзакцию и проверяет, что эта транзакция непротиворечива (все необходимые сервисы могут быть активированы независимо друг от друга).
- **Активация сервисов ОС по требованию**. В отличие от SystemD при использовании менеджера сервисов SysVinit активируются все возможные процессы фоновых сервисов, которые в принципе могут использоваться. Пользователи могут войти в систему только после активации всех этих сервисных процессов прописанных в выбранном уровне загрузке операционной системы (Runlevel). В результате недостатки SysVinit очевидны: медленная загрузка системы. И это при том, что некоторые сервисы заявленные в Runlevel могут практически никогда не использоваться во время работы системы или использоваться только эпизодически и не требуют постоянной активности. В качестве примера такого сервиса можно привести сервер печати CUPS, который на большинстве серверов используются эпизодически и требует активности только в момент нажатия пользователем кнопки “Печать” в каком-либо прикладном ПО. На многих серверах редко используется SSHD, чья активность по сути нужна только в момент когда к данному сервису идет удаленное обращение от пользователя. Тратить время на запуск и постоянную активность этих сервисов и системных ресурсов бессмысленно. Активация systemd возможна только при запросе сервиса. Если запрос завершен, systemd останавливает работу.

Совместимость со скриптами инициализации сервисов SysVinit

Как и в UpStart, в systemd вводятся новые методы настройки и предъявляются новые требования к разработке приложений. Если вы хотите заменить текущую систему инициализации на systemd, то нужно обеспечить совместимость systemd с существующей программой. В любом дистрибутиве Linux изменить весь код сервисов за короткое время

для использования systemd будет непросто.

В systemd доступны функции, совместимые со скриптами инициализации SysVinit и LSB. Изменять имеющиеся в системе сервисы и процессы не требуется. Это уменьшает цену переноса системы на systemd, позволяя пользователям заменить на systemd имеющуюся систему инициализации.

Совместимость команд SysVinit и systemd

Функции команды systemctl из systemd подобны функциям команды SysVinit. Обратите внимание, что в этой версии поддерживаются команды service и chkconfig. Тем не менее, настоятельно рекомендуется использовать команду systemctl для управления всеми системными сервисами.

В таблице 17 представлено сравнение команд менеджеров сервисов SystemD и SysVinit

Таблица 17. Сравнение ключевых команд менеджеров сервисов SystemD и SysVinit

Команда SysVinit	Команда systemd	Описание
service network start	systemctl start network.service	Запускает сервис
service network stop	systemctl stop network.service	Останавливает сервис
service network restart	systemctl restart network.service	Перезапускает сервис
service network reload	systemctl reload network.service	Перезагружает файл конфигурации без прерывания операции
service network condrestart	systemctl condrestart network.service	Перезапускает сервис, только если он работает
service network status	systemctl status network.service	Проверяет состояние работы сервиса
chkconfig network on	systemctl enable network.service	Включает сервис, когда наступает время его активации или выполняется условие триггера для его включения

Команда SysVinit	Команда systemd	Описание
chkconfig network off	systemctl disable network.service	Отключает сервис, когда наступает время его активации или выполняется условие триггера для его отключения
chkconfig network	systemctl is-enabled network.service	Проверяет, включен ли сервис
chkconfig --list	systemctl list-unit-files --type=service	Выводит список всех сервисов на каждом уровне выполнения и проверяет, включены ли они
chkconfig network --list	ls /etc/systemd/system/*.wants/network.service	Перечисляет уровни выполнения, на которых включен сервис, и те, на которых он отключен
chkconfig network --add	systemctl daemon-reload	Используется, когда нужно создать сервисный файл или изменить настройки

Управление жизненным циклом сервисов с помощью Cgroups

Важной ролью менеджера сервисов ОС является отслеживание и управление их жизненным циклом. Он может запускать и останавливать сервис. Однако закодировать систему инициализации для остановки сервисов гораздо сложнее, чем можно подумать. Сервисные процессы часто работают в фоновом режиме как демоны и иногда дважды разветвляются. Для UpStart необходима корректная настройка строфы expect в файле конфигурации. В противном случае UpStart не сможет узнать ИД процесса (PID) демона, подсчитав количество ответвлений.

Контрольные группы (Cgroups), давно используемые для управления квотами системных ресурсов, упрощают эти задачи. Их простота во многом обусловлена пользовательским интерфейсом, аналогичным файловой системе. Когда родительский сервис создает дочерний, последний наследует все атрибуты группы Cgroup, в которую входит первый. Это означает, что все необходимые сервисы помещаются в одну и ту же Cgroup. Для systemd достаточно просто пройтись по контрольной группе, чтобы найти PID всех нужных сервисов, а затем остановить их один за другим.

Управление точками монтирования и автоматического монтирования

В традиционных системах Linux пользователи могут поддерживать фиксированные точки монтирования файловой системы с помощью файла /etc/fstab. Эти точки автоматически монтируются при запуске системы. После запуска они становятся доступными в указанных каталогах. Эти точки монтирования являются файловыми системами, которые критически важны для работы всей системы, например домашние каталоги пользователей HOME. Как и SysVinit, systemd контролирует эти точки, чтобы они автоматически монтировались при запуске системы. И systemd также обеспечивает совместимость с файлом /etc/fstab. Пользователь как и прежде может использовать его для управления точками монтирования.

Бывают случаи, когда необходимы монтирование или размонтирование по требованию. Например, временная точка монтирования необходима для доступа к содержимому на сменных носителях данных таких как DVD-диск, подключаемые флеш-диски и пр. А когда оно больше не нужно, точка монтирования удаляется (с помощью команды umount) для экономии ресурсов. Традиционно это делается с помощью сервиса autofs.

При использовании systemd возможно автоматическое монтирование без необходимости установки autofs.

Моментальные снимки и восстановление состояния системы

Запускать systemd можно по запросу. Из-за этого рабочее состояние системы изменяется динамически и администратор не может знать, какие конкретно сервисы в системе сейчас выполняются а какие нет. Моментальные снимки systemd позволяют сохранять и восстанавливать текущее состояние работы системы.

Например, если в системе запущены сервисы A и B, пользователь может выполнить команду systemd, чтобы создать моментальный снимок ее текущего состояния. Затем остановите процесс A или внесите в систему любое другое изменение, например запустите процесс C. После этого выполните команду systemd для восстановления из снимка, чтобы восстановить систему до точки, в которой был сделан снимок, то есть когда работали только сервисы A и B. Возможный сценарий применения — отладка. Например, если на сервере возникает исключение, пользователь может сохранить текущее состояние в виде моментального снимка для целей отладки и выполнить любую операцию, например, остановить сервис. После завершения отладки происходит восстановление из снимка.

Вывод информации о существующих сервисах системы

Для того чтобы ознакомиться со списком всех загруженных в данный момент сервисов, выполните следующую команду:

systemctl list-units --type service

Для того чтобы ознакомиться со списком всех сервисов систем вне зависимости от того, загружены они в данный момент времени или нет, выполните следующую команду (с параметром all):

systemctl list-units --type service --all

Пример списка всех загруженных на данный момент сервисов:

```
$ systemctl list-units --type service
UNIT                      LOAD   ACTIVE     SUB     JOB  DESCRIPTION
atd.service               loaded active     running       Deferred execution scheduler
auditd.service            loaded active     running       Security Auditing Service
avahi-daemon.service      loaded active     running       Avahi mDNS/DNS-SD Stack
chronyd.service           loaded active     running       NTP client/server
crond.service             loaded active     running       Command Scheduler
dbus.service              loaded active     running       D-Bus System Message Bus
dracut-shutdown.service   loaded active     exited        Restore /run/initramfs on
shutdown
firewalld.service         loaded active     running       firewalld - dynamic firewall daemon
getty@tty1.service        loaded active     running       Getty on tty1
gssproxy.service          loaded active     running       GSSAPI Proxy Daemon
irqbalance.service        loaded active     running       irqbalance daemon
iscsid.service            loaded activating start   start Open-iSCSI
```

В таблице 18 представлены параметры в выходных данных команды которые пользователь может увидеть в выводе данной команды.

Таблица 18. Параметры сервисов ОС

Параметр	Описание
Loaded	Информация о том, загружен ли сервис, абсолютный путь к файлу сервиса и запись о том, включен ли он
Active	Информация о том, запущен ли сервис, и метка времени
Main PID	PID сервиса
Cgroup	Дополнительная информация о связанных контрольных группах

Отображение текущего статуса сервисов ОС

Чтобы отобразить состояние сервиса, выполните следующую команду:

systemctl status <имя>.service

Чтобы проверить, запущен ли конкретный сервис, выполните следующую команду:

systemctl is-active <имя>.service

Пояснение выводимых опцией is-active данных представлено в таблице 19.

Таблица 19. Данные выводимые опцией is-active

Состояние	Описание
active(running) active(exited)	Сервис в системе работает. Сервис, штатно завершающий работу после срабатывания только один раз. В настоящее время никакая программа в системе не запущена. Например, функция quotaon выполняется только при запуске или монтировании программы.
active(waiting)	Программа ожидает других событий, чтобы продолжить работу. Например, сервис очереди печати запускается, но ему необходимо получить что-то в очередь (задания на печать), чтобы затем пробудить сервис принтера, который выполнит следующую функцию печати.
inactive	Сервис не запущен.

Аналогично вышеописанному, чтобы определить, включен ли конкретный сервис, выполните необходимо выполнить представленную ниже команду:
systemctl is-enabled <имя>.service

Пояснение выводимых опцией is-enabled данных представлено в таблице 20.

Таблица 20. Данные о сервисе выводимые опцией is-enabled

Состояние	Описание
“enabled”	Сервис постоянно включен через псевдоним Alias=, .wants/ или символьную ссылку .requires/ в каталоге /etc/systemd/system/.
“enabled-runtime”	Сервис временно включен через псевдоним Alias=, .wants/ или символьную ссылку .requires/ в каталоге /run/systemd/system/.
“linked”	Хотя файл компонента не находится в стандартном каталоге компонентов, на него имеются символьные ссылки в постоянном каталоге /etc/systemd/system/.
“linked-runtime”	Хотя файл компонента не находится в стандартном каталоге компонентов, на него имеются символьные ссылки во временном каталоге /run/systemd/system/.
“masked”	Сервис постоянно замаскирован в каталоге /etc/systemd/system/ (символьная ссылка на /dev/null). Из-за этого операция start не срабатывает.
“masked-runtime”	Сервис временно замаскирован в каталоге /run/systemd/system/ (символьная ссылка на /dev/null). Из-за этого операция start не срабатывает.
“static”	Сервис не включен. В разделе [Install] файла компонента нет доступного параметра для команды enable.
“indirect”	Сервис не включен. Однако список значений для параметра Also= в разделе [Install] файла компонента не пустой (то есть некоторые компоненты в списке могли быть включены) либо файл компонента имеет символьную ссылку-псевдоним, которая не входит в список Also=. Для компонента шаблона это означает, что включен экземпляр, отличный от DefaultInstance=.

Состояние	Описание
“disabled”	Сервис не включен. Однако раздел [Install] файла компонента содержит параметры, доступные для команды enable.
“generated”	Файл компонента создается генератором автоматически. Сгенерированный файл не может быть включен напрямую, но неявно включается генератором.
“transient”	Файл компонента создается динамически и на время API-интерфейсом runtime (среды выполнения). Включение временного компонента невозможно.
“bad”	Файл компонента неверный, или возникают другие ошибки. Команда is-enabled не возвращает это состояние, а выдает сообщение об ошибке. Этот компонент может быть отображен командой list-unit-files.

К примеру, чтобы отобразить состояние сервиса GDM (gdm.service), необходимо выполнить следующую команду

systemctl status gdm.service.

 # systemctl status gdm.service
gdm.service - GNOME Display Manager Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled) Active: active (running) since Thu 2023-10-17 17:31:23 CEST; 5min ago
Main PID: 1029 (gdm)
CGroup: /system.slice/gdm.service
1029 /usr/sbin/gdm
1037 /usr/libexec/gdm-simple-slave --display-id /org/gno...
1047 /usr/bin/Xorg :0 -background none -verbose-auth /r...Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.

Запуск сервиса ОС

Для того чтобы запустить сервис, необходимо выполнить команду представленную ниже. Она доступна только суперпользователю (root):

systemctl start name.service

К примеру, запустим сервис отвечающий за веб-сервер

[root@host /]# systemctl start httpd.service

Остановка сервиса

Для того чтобы остановить сервис необходимо выполнить следующую команду доступную суперпользователю (root):

systemctl stop <имя>.service

К примеру, для остановки сервиса обеспечивающего Bluetooth подключения, выполните следующую команду доступную суперпользователю (root):

 # systemctl stop bluetooth.service

Перезапуск сервиса

Для того чтобы перезапустить сервис, выполните следующую команду доступную суперпользователю (root):

systemctl restart <имя>.service

Данная команда останавливает выбранный сервис в текущем сеансе и сразу же запускает его снова. Если выбранный сервис не выполняется, то данный сервис просто будет запущен.

К примеру, чтобы перезапустить сервис обеспечивающий Bluetooth подключения, выполните следующую команду доступную суперпользователю (root):

 # systemctl restart bluetooth.service

Включение сервиса в автозагрузку

Для того чтобы настроить автоматический запуск сервиса во время загрузки системы, выполните следующую команду от имени суперпользователя root:

systemctl enable <имя>.service

К примеру, чтобы настроить при загрузке системы автоматический запуск веб-сервера httpd, выполните следующую команду:

 # systemctl enable httpd.service

ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'

Исключение сервиса из автозагрузки

Для того чтобы сервис не запускался во время загрузки системы автоматически, от имени суперпользователя root необходимо выполнить следующую команду:

systemctl disable <имя>.service

Например, чтобы предотвратить при загрузке системы автоматический запуск сервиса обеспечения связи по Bluetooth, выполните следующую команду:

```
# systemctl disable bluetooth.service
Removed /etc/systemd/system/bluetooth.target.wants/bluetooth.service.
Removed /etc/systemd/system/dbus-org.bluez.service.
```

Уровни выполнения и цели

В менеджере сервисов ОС systemd понятие уровней выполнения заменено на “цели” для большей гибкости. Например, вы можете наследовать существующую цель и преобразовать ее в цель для своих нужд, добавив другие сервисы. В таблице 21 представлен полный список уровней выполнения и соответствующих им целей systemd.

Таблица 21. Соответствие целей и уровней исполнения ОС

Уровень выполнения	Цель systemd	Описание
0	runlevel0.target, poweroff.target	Выключение операционной системы
1, s, single	runlevel1.target, rescue.target	Операционная система работает в однопользовательском режиме.
2, 4	runlevel2.target, runlevel4.target, multi-user.target	Операционная система находится на уровне выполнения, который задан пользователем или относится к конкретному домену (по умолчанию эквивалентно уровню выполнения 3).
3	runlevel3.target, multi-user.target	Операционная система находится в неграфическом многопользовательском режиме и доступна из нескольких консолей или сетей.
5	runlevel5.target, graphical.target	Операционная система находится в графическом многопользовательском режиме. Все сервисы, работающие на уровне 3, доступны для входа через графический интерфейс.
6	runlevel6.target, reboot.target	Операционная система перезагружается.
emergency	emergency.target	Аварийная оболочка.

Просмотр цели запуска по умолчанию

Выполните следующую команду, чтобы просмотреть цель запуска системы по умолчанию:

systemctl get-default

Просмотр всех доступных в системе целей запуска

Выполните следующую команду, чтобы просмотреть все цели запуска системы:

systemctl list-units --type=target

Изменение цели загружаемой по умолчанию

Для того чтобы изменить цель по умолчанию, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl set-default <имя>.target

Изменение текущей цели

Для того чтобы изменить текущую цель, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl isolate <имя>.target

Переход в режим восстановления

Для того чтобы перевести операционную систему в режим восстановления, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl rescue

Данная команда аналогична команде **systemctl isolate rescue.target**. После выполнения команды на последовательный порт выводится следующая информация.

You are in rescue mode. After logging in, type "journalctl -xb" to viewsystem logs, "systemctl reboot" to reboot, "systemctl default" or "exit"to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):

Важно помнить, что чтобы перейти из режима восстановления в нормальный режим работы, необходимо перезагрузить систему.

Переход в аварийный режим

Для того чтобы перевести операционную систему в аварийный режим, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl emergency

Данная команда аналогична команде systemctl isolate emergency.target. После выполнения команды на последовательный порт выводится следующая информация.

You are in emergency mode. After logging in, type "journalctl -xb" to viewsystem logs, "systemctl reboot" to reboot, "systemctl default" or "exit"to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):

Важно помнить, что чтобы перейти из аварийного режима в нормальный режим работы, необходимо перезагрузить систему.

6.5.1 Базовое управление питанием ОС

Вместо старых команд управления системой в systemd используется команда **systemctl** для завершения работы, перезапуска, приостановки и активации спящего режима операционной системы. Хотя предыдущие команды управления системой по-прежнему доступны в systemd из соображений совместимости, рекомендуется использовать systemctl, когда это возможно. Соответствие между ними показано в таблице 22.

Таблица 22. Соответствие команд управления питанием SystemD классическим командам

Классическая команда управления питанием	Команда systemctl	Описание
halt	systemctl halt	Завершает работу операционной системы.
poweroff	systemctl poweroff	Выключает операционную систему.
reboot	systemctl reboot	Перезагружает операционную систему.

Завершение работы операционной системы

Для того чтобы завершить работу операционной системы и выключить систему, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl poweroff

Для того чтобы завершить работу операционной системы, не выключая ее, необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl halt

По умолчанию при выполнении любой из этих команд systemd отправляет информационное сообщение всем пользователям, вошедшим в систему о завершении работы системы. Чтобы это сообщение systemd не отправлялось, выполните команду с опцией**–no-wall**. Команда выглядит следующим образом:

systemctl --no-wall poweroff

Перезагрузка операционной системы

Для того чтобы перезагрузить операционную систему,необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl reboot

По умолчанию при выполнении любой из этих команд systemd отправляет информационное сообщение всем пользователям, вошедшим в систему. Чтобы это сообщение systemd не отправлялось, выполните команду с опцией **–no-wall**. Команда выглядит следующим образом:

systemctl --no-wall reboot

Приостановка операционной системы

Для того чтобы приостановить работу операционной системы,необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl suspend

Гибернация операционной системы

Для того чтобы перевести операционную систему в режим гибернации,необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl hibernate

Чтобы приостановить работу операционной системы и перевести ее в режим гибернации,необходимо от имени суперпользователя (root) выполнить следующую команду:

systemctl hybrid-sleep

6.6 ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

- В чем преимущества использования SystemD в сравнении с SysVinit и Upstart?
- В чем преимущества и недостатки использования бинарных RPM пакетов для установки программного обеспечения?
- Скачайте исходные коды веб-сервера nginx и проведите их сборку и последующую установку в систему.

7. УПРАВЛЕНИЕ ПОДСИСТЕМОЙ ХРАНЕНИЯ И ФАЙЛОВЫМИ СИСТЕМАМИ

Как мы ранее уже обсуждали в предыдущих главах, все дистрибутивы Linux исповедуют тезис “Все это файл”. Таким образом, в виде файлов в операционной системе представляются не только пользовательские данные но даже и аппаратные устройства.

Для управления дисковой подсистемой хранения данных в дистрибутиве OpenScaler стандартом де-факто является использование LVM (Logical Volume Manager).

LVM — это дополнительный слой абстракции от железа, позволяющий собрать произвольное количество разнородных дисков в один единых пул хранения данных и оперировать им не на уровне дисков, а на уровне доступного объема хранения данных, и при необходимости разбить его на один или несколько разделов под разные точки монтирования в системе.

LVM в основе своей использует 3 уровня абстракции:

- Физический уровень (**PV**). Сначала диск инициализируется командой **pvcreate** — в начале диска создается дескриптор группы томов. При этом важно заметить, что диск не обязательно должен быть физическим — можно отметить на использование обычный раздел диска.
- Группа томов (**VG**). С помощью команды **vgcreate** создается группа томов из инициализированных на предыдущем этапе дисков.
- Логический том (**LV**). Группы томов нарезаются на логические тома командой **lvcreate**.

Таким образом, цепочку абстракции можно представить следующей: Физический жесткий диск (или иное устройство хранения) Группа томов (VG) как единое доступное пространство для хранения данных собранное из всех доступных физических дисков Логический том (LV) фактически виртуальный раздел обладающий своей файловой системой и доступной для монтирования в системе как самостоятельное устройство хранения данных.

Схематически это представлено на рисунке 21.

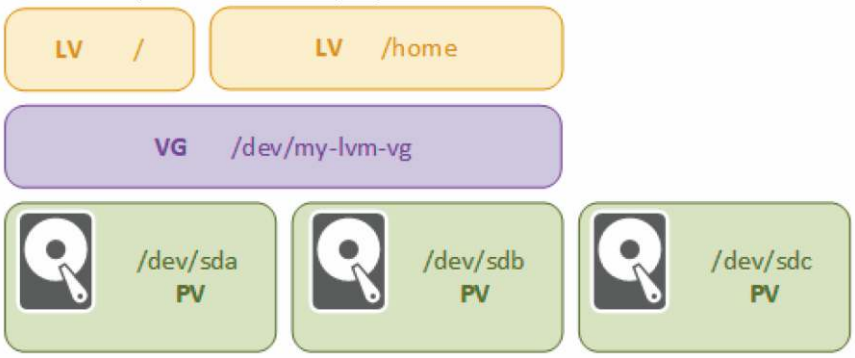

Рисунок 21. Уровни абстракции LVM

Базовые принципы работы с подсистемой хранения

Физические устройства хранения обычно представляются в системе файлами устройств типа /dev/hdx (для IDE устройств) и /dev/sdx (для SAS/SATA устройств), где x - это буква от a до z однозначно идентифицирующая диск. (sda, hdc)

Представлять жёсткий диск как единый «лист» не всегда бывает удобно: иногда полезно «разрезать» его на несколько независимых листов, на каждом из которых можно писать и стирать что угодно, не опасаясь повредить написанное на других листах. Для этого в системе вводится понятие раздела (partition).

Партиция - по сути, область жесткого диска определенная границами секторов представляющаяся системе как отдельное устройство хранения данных. В системе эти партиции номеруются и представляются в системе как устройства /dev/sdx1, 2, 3, 4, где x - определяет диск (sda), а последующая цифра - номер раздела на данном жестком диске.

С точки зрения уровня абстракции LVM:

- **PV**, Physical volume, физический том. Обычно это раздел на диске или весь диск. В том числе, устройства программного и аппаратного RAID (которые уже могут включать в себя несколько физических дисков). Физические тома входят в состав группы томов.
- **VG**, Volume group, группа томов. Это самый верхний уровень абстрактной модели, используемой системой LVM. С одной стороны группа томов состоит из физических томов, с другой -- из логических и представляет собой единую административную единицу.
- **LV**, Logical volume, логический том. Раздел группы томов, эквивалентен разделу диска в не-LVM системе. Представляет собой блочное устройство и, как следствие, может содержать файловую систему.
- **PE**, Physical extent, физический экстент. Каждый физический том делиться на порции данных, называющиеся физическими экстентами. Их размеры те же, что и у логических экстентов.
- **LE**, Logical extent, логический экстент. Каждый логический том делится на порции данных, называющиеся логическими экстентами. Размер логических экстентов не меняется в пределах группы томов.

7.1 УСТАНОВКА И РАБОТА С LOGICAL VOLUME MANAGER

7.1.1 Установка LVM

Logical Volume Manager является стандартным пакетом OpenScaler и входит в состав базовой установки ОС. Вы можете выполнить представленную ниже команду чтобы в этом удостовериться а заодно и получить описание о текущей версии установленного пакета

```
rpm -qa | grep lvm2
```

Тем не менее, если по какой-то причине пакет LVM был удален из системы и вышеуказанная команда не нашла данный пакет в числе установленных, вы можете самостоятельно провести его установку выполнив следующую команду

```
[root@host /]# dnf install lvm2
```

7.1.2 Создание физического раздела на устройстве хранения

В случае использования нового, неразмеченного жесткого диска, в базовом наиболее простом случае процесс создания физического раздела представляет из себя следующую последовательность действий. Для создания разделов могут использоваться различные программные инструменты такие как parted, fdisk и прочие. В рамках данного примера будем использовать инструмент fdisk.

Запустим fdisk указав в качестве параметра имя желаемого физического диска

```
[root@host -]# fdisk /dev/sdb
```

Находясь в командной строке fdisk, выполним следующую последовательность действий:

- введем **n** - создание нового раздела
- выберем порядковым номер создаваемого раздела (при пустом диске это 1)
- укажем начальные и конечные границы нового раздела - в секторах или ГБ дискового пространства. Но поскольку мы использовать будем размер всего диска для единого раздела, то на вопросы о начальной и конечной границы раздела достаточно просто нажать “ввод”, не вводя никаких значений
- введем **l** - чтобы выбрать тип раздела в зависимости от его назначения
- введем **8e** - чтобы указать тип раздела LVM
- введем **w** - чтобы сохранить новый раздел и его настройки.

В оболочке ОС это выглядит следующим образом:

```
[root@host -]# fdisk /dev/sdb
Command (m for help): n # Создаем новый раздел
Partition type
p primary (0 primary, 0 extended, 4 free)
e extended (container for logical partitions)
Select (default p): p # Выбираем физический раздел
Partition number (1-4, default 1): # Нажимаем Ввод.
First sector (2048-10485759, default 2048): #Нажимаем Ввод.
Last sector, +sectors or +size{K,M,G,T,P} (2048-10485759, default 10485759): # Нажимаем Ввод.
Created a new partition 1 of type'Linux' and of size 5 GiB.
Command (m for help): t #Смена типа раздела
Selected partition 1
Hex code (type L to list all codes): 8e# Выбираем LVM Type
Changed type of partition'Linux' to'Linux LVM'.
Command (m for help): w# Сохраняем конфигурацию
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Далее можно ввести команду **fdisk -l**, дабы удостовериться что требуемый раздел создан и доступен.

```
Calling ioctl() to re-read partition table.
Syncing disks.
[root@host ~]# fdisk -l
Disk /dev/vda: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x16a1e057

Device Boot Start End Sectors Size Id Type
/dev/sda1 * 2048 20971486 20969439 10G 83 Linux

Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x03864530

Device Boot Start End Sectors Size Id Type /dev/sdb1 2048 10485759 10483712 5G 8e Linux LVM
```

7.1.3 Создание PV (Physical Volume)

Для создания PV используется команда **pvcreate**. Ее синтаксис представлен ниже
pvcreate [опции] имя_устройства

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-f** - создание без какого-либо подтверждения.
- **-u** - указать UUID устройства
- **-y** - ответить да на все вопросы

Примеры наиболее распространенных случаев использования команды:

- Создать PV из /dev/sdb и /dev/sdc
  - **[root@host /]# pvcreate /dev/sdb /dev/sdc**
- Создать PV из /dev/sdb1 и /dev/sdb2
  - **[root@host /]# pvcreate /dev/sdb1 /dev/sdb2**

7.1.4 Просмотр данных о PV (Physical Volume)

Для просмотра данных о PV используется команда **pvdisplay**. Ее синтаксис представлен ниже :
pvdisplay [опции] имя_устройства

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-s** - выводить данные в укороченном формате
- **-m** - отобразить соотношение между PE и LE.

Примеры наиболее распространенных случаев использования команды:

- Просмотреть информацию о PV
  - **pvdisplay /dev/sdb**

7.1.5 Изменение атрибутов PV (Physical Volume)

Для изменения атрибутов PV используется команда **pvchange**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.
pvchange [опции] имя_PV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-u** - сгенерировать новый UUID
- **-x** - определяет разрешена ли аллокация PE

Примеры наиболее распространенных случаев использования команды:

- Запретить PE для /dev/sdb
  - **# pvchange -x n /dev/sdb**

7.1.6 Удаление PV (Physical Volume)

Для удаления PV используется команда **pvremove**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.

pvremove [опции] имя_PV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-f** - принудительно удалить PV без дополнительного подтверждения пользователем
- **-y** - отвечать утвердительно на все вопросы задаваемые в интерактивном режиме

Примеры наиболее распространенных случаев использования команды:

- Удалить PV /dev/sdb
  - **# pvremove /dev/sdb**

7.1.7 Создание VG (Volume Group)

Для создания VG используется команда **vgcreate**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgcreate [опции] имя_VG имя_PV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-l** - указать максимальное количество LV в одной VG
- **-p** - указать максимальное количество PV в одной VG
- **-s** - указать размер PE для PV

Примеры наиболее распространенных случаев использования команды:

- Создать VG1 из PV /dev/sdb и /dev/sdc
  - **[root@host /]# vgcreate vg1 /dev/sdb /dev/sdc**

7.1.8 Просмотр VG (Volume Group)

Для просмотра VG используется команда **vgdisplay**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgdisplay [опции] имя_VG

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-A** - показывать атрибуты только активной VG
- **-s** - выводить данные в укороченном формате

Примеры наиболее распространенных случаев использования команды:

- Отобразить базовую информацию о VG1
  - **[root@host /]# vgdisplay vg1**

7.1.9 Изменение атрибутов VG (Volume Group)

Для изменения атрибутов VG используется команда **vgchange** доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgchange [опции] имя_VG

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- -a - установить “активный” статус для VG

Примеры наиболее распространенных случаев использования команды:

- Установить активный статус для VG1
  - [root@host /]# vgchange -ay vg1

7.1.10 Увеличение размера VG (Volume Group)

Размер группы томов может быть изменен добавлением в него новых физических томов для использования. Данную задачу также может осуществлять исключительно суперпользователь (root).

Для этого используется команда **vgextend** доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgextend [опции] имя_VG имя_PV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-d** - исполнить команду в режиме отладки
- **-t** - только протестировать действие, не внося реальных изменений
- **имя_VG** - имя VG которую планируется увеличить
- **имя_PV** - имя PV которую планируется включить в VG

Примеры наиболее распространенных случаев использования команды:

- Расширить VG1 путем добавление в нее PV - /dev/sdb
  - **[root@host /]# vgextend vg1 /dev/sdb**

7.1.11 Уменьшение размера VG (Volume Group)

Размер группы томов может быть уменьшен за счет исключения из него ранее добавленных физических томов для использования. Данную задачу также может осуществлять исключительно суперпользователь (root). Также важно помнить что удалить последний оставшийся PV из группы томов невозможно.

Для этого используется команда vgreduce доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgreduce [опции] имя_VG имя_PV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-a** - Если не указано имя PV удалить все незадействованные PV
- **--removemissing** - Удалить потерянные, нерабочие PV восстановив тем самым нормальное состояние VG
- **имя_VG** - имя VG которую планируется уменьшить
- **имя_PV** - имя PV которую планируется исключить из VG

Примеры наиболее распространенных случаев использования команды:

- Уменьшить VG1 путем удаления из нее PV - /dev/sdb2
  - **[root@host /]# vgreduce vg1 /dev/sdb2**

7.1.12 Удаление VG (Volume Group)

Для этого используется команда **vgremove**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.

vgremove [опции] имя_VG

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-f** - принудительно удалить VG без дополнительного подтверждения пользователем
- **имя_VG** - имя VG которую планируется удалить

Примеры наиболее распространенных случаев использования команды:

- Удалить том VG1
  - **[root@host /]# vgremove vg1**

7.1.13 Создание LV (Logical Volume)

Для создания LV используется команда **lvcreate**, доступная только суперпользователю (root). Ее синтаксис представлен ниже.

lvcreate [опции] имя_VG

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- -L - указать размер LV (в мб, гб)
- -l - указать размер LV в PE
- -n - указать имя создаваемой LV

Примеры наиболее распространенных случаев использования команды:

- Создать LV размером 10GB на VG1
  - **[root@host /]# lvcreate -L 10G vg1**
- Создать LV размером 200MB на VG1 и назвать lv1
  - **[root@host /]# lvcreate -L 200M -n lv1 vg1**

7.1.14 Просмотр данных о LV (Logical Volume)

Для просмотра данных о LV используется команда **lvdisplay**. Ее синтаксис представлен ниже

lvdisplay [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-v** - Отображать аллокацию между LE и PE

Важно знать, если LV создается в определенном томе VG то она располагается в каталоге представляющем данный том. К примеру, если создать lv1 на томе vg1 то ее расположение будет **/dev/vg1/lv1**

Примеры наиболее распространенных случаев использования команды:

- Просмотреть информацию о LV
  - **# lvdisplay /dev/vg1/lv1**

7.1.15 Изменение размера LV (Logical Volume)

Для изменения размера LV используется команда **lvresize**. Ее синтаксис представлен ниже.

lvresize [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-L** - указать размер LV (в мб, гб)
- **-l** - указать размер LV в PE
- **-f** - принудительно изменить размер LV без дополнительного подтверждения пользователя

Важно знать, при изменении размера LV нужно действовать осторожно при уменьшении размера, при некорректно выставленном размере есть риск потери хранимых на LV данных.

Примеры наиболее распространенных случаев использования команды:

- Добавить 200МБ в LV
  - lvresize -L +200 /dev/vg1/lv1
- Уменьшить LV на 200МБ
  - # lvresize -L -200 /dev/vg1/lv1

7.1.16 Увеличение размера LV (Logical Volume)

Для увеличения размера LV используется команда **lvextend.** Ее синтаксис представлен ниже

lvrextend [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-L** - указать размер LV (в мб, гб)
- **-l** - указать размер LV в PE
- **-f** - принудительно изменить размер LV без дополнительного подтверждения пользователя

Примеры наиболее распространенных случаев использования команды:

- Добавить 100МБ в LV
  - **[root@host /]# lvextend -L +100M /dev/vg1/lv1**

7.1.17 Уменьшение размера LV (Logical Volume)

Для уменьшения размера LV используется команда **lvreduce**. Ее синтаксис представлен ниже.

lvreduce [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- -L - указать размер LV (в мб, гб)
- -l - указать размер LV в PE
- -f - принудительно изменить размер LV без дополнительного подтверждения пользователя

Важно знать, при уменьшении размера LV нужно действовать осторожно при уменьшении размера, при некорректно выставленном размере есть риск потери хранимых на LV данных.

Примеры наиболее распространенных случаев использования команды:

- Уменьшить LV на 100МБ
  - **[root@host /]# lvreduce -L -100M /dev/vg1/lv1**

7.1.18 Удаление LV (Logical Volume)

Для удаления LV используется команда **lvremove**. Ее синтаксис представлен ниже

lvremove [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-f** - принудительно удалит LV без дополнительного подтверждения пользователя

Важно знать, удаление LV возможно осуществить только при отмонтированной файловой системе на ней.

Примеры наиболее распространенных случаев использования команды:

- Удалить LV1
  - **[root@host /]# lvremove /dev/vg1/lv1**

7.2 СОЗДАНИЕ И МОНТИРОВАНИЕ ФАЙЛОВЫХ СИСТЕМ

После того как вы создали LV - последнее что остается сделать это создать на ней требуемую файловую систему и смонтировать ФС в требуемую точку монтирования.

7.2.1 Создание файловых систем

Для создания файловой системы используется команда **mkfs**. Ее синтаксис представлен ниже.

mkfs [опции] имя_LV

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-t** - указывает тип создаваемой файловой системы (ext2,3,4)

Примеры наиболее распространенных случаев использования команды:

- Создание файловой системы ext4 на LV1
  - **[root@host /]# mkfs -t ext4 /dev/vg1/lv1**

7.2.2 Монтирование ФС в ручном режиме

Файловые системы, вручную смонтированные администратором, будут доступны только до момента перезагрузки или выключения системы. После перезагрузки файловые системы придется еще раз смонтировать самостоятельно.

Для монтирования файловой системы используется команда **mount**. Ее синтаксис представлен ниже

mount [опции] имя_LV имя_точки_монтирования

Под точкой монтирования понимается каталог, к корню которого будет смонтирована новая файловая система.

Примеры наиболее распространенных случаев использования команды:

- Смонтировать файловую систему на LV1 в каталог /mnt/data
  - **[root@host /]# mount /dev/vg1/lv1 /mnt/data**

7.2.3 Автоматическое монтирование ФС

Процесс монтирования всех требуемых файловых систем может быть автоматизирован и осуществляться при загрузке операционной системы без фактического вмешательства пользователя и каких бы то ни было ручных операций.

Для настройки автоматического монтирования файловых систем необходимо выполнить следующие действия.

Выполните команду **blkid** для того чтобы узнать UUID для LV1

[root@host /]# blkid /dev/vg1/lv1

Вывод команды выглядит следующим образом

/dev/vg1/lv1: UUID="uuidnumber" TYPE="fstype"

Отредактируйте файл **/etc/fstab**, используя для этого редактор vi, описанный в предыдущих главах и добавьте в него следующую строку:

UUID=uuidnumber mntpath fstype defaults 0 0

В данной строке:

- **UUID** - идентификатор LV1 который мы узнали, выполнив blkid
- **mntpath** - точка монтирования, каталог, в который должна быть смонтирована файловая система на LV1
- **fstype** - тип используемой файловой системы, который мы также узнали, выполнив blkid
- **defaults** - дополнительные параметры монтирования файловой системы.
- **0** - нужно ли производить резервное копирование файловой системы, по умолчанию нет
- **0** - нужно ли проверять файловую систему на наличие ошибок при перезапуске системы, по умолчанию нет

Приведем пример настройки автоматического монтирования

- Отмонтируем LV1 если она уже была смонтирована
  - **[root@host /]# umount /dev/vg1/lv1**
- Выполним следующую команду, чтобы перечитать конфигурационный файл /etc/fstab
  - **[root@host /]# mount -a**
- Проверим смонтировалась ли файловая система (точка монтирования указана в качестве примера)
  - **[root@host /]# mount | grep /mnt/data**
- Проверьте вывод команды выше на наличие строки **/dev/vg1/lv1 on /mnt/data**. Если она присутствует, файловая система успешно смонтировалась автоматически.

7.3 Вопросы для самопроверки

1. Какие уровни абстракции добавляет использование LVM?

2. Используя LVM создайте том и логический раздел с 100LE и размером PE равным 8мб

3. Создайте файловую систему на логическом разделе и настройте ее автомонтирование в каталог /mnt/data

8. КЛЮЧЕВЫЕ АДМИНИСТРАТИВНЫЕ ОПЕРАЦИИ

8.1 АВТОМАТИЗАЦИЯ ВЫПОЛНЕНИЯ ЗАДАНИЙ

Иногда отдельные задачи по администрированию операционной системы приходится проводить с завидной периодичностью, что превращается в рутину для системного администратора. Выходом из этой ситуации является возможность автоматизации исполнения определенных задач и действий, используя инструментарий, доступный в системе.

8.1.1 Единичное исполнение команд (AT)

Команда **at** используется для назначения одноразового задания на заданное время. Чтобы назначить выполнение разового задания в определённое время, введите команду at time, где time — время выполнения команды.

Параметр time может быть следующим:

- формат ЧЧ:ММ — например, 04:00 обозначает четыре часа ночи. Если указанное время уже прошло, задание выполняется в это же время на следующий день.
- midnight — обозначает полночь.
- noon — обозначает полдень.
- teatime — обозначает 4 часа дня (время чаепития).
- формат «название-месяца день год» — например, строка «January 15 2024» обозначает 15 января 2024 года. Указывать год не обязательно.
- Форматы ММДДГГ, ММ/ДД/ГГ или ММ.ДД.ГГ — например, 011502 также обозначает 15 января 2002 г.
- now + время — время задаётся в минутах, часах, днях или неделях. Например, строка «now + 5 days» обозначает, что команда должна быть выполнена в это же время, но через пять дней.

В директории /etc также могут присутствовать два файла регламентирующие поведение команды.Один из них это блок-лист /etc/at.deny и “доверенный лист” /etc/at.allow позволяющие запрещать или разрешать использование команды определенным пользователям системы.

- Если в системе присутствует файл /etc/at.allow - то пользоваться at смогут только пользователи в нем перечисленные. При этом файл /etc/at.deny будет проигнорирован даже если он существует
- Если в системе присутствует файл /etc/at.deny но нет файла at.allow- то все пользователи системы могут пользоваться функционалом at, кроме тех что перечислены в файле at.deny
- Если оба файла в системе отсутствуют то пользоваться at имеет право только суперпользователь (root)

Пример использования команды:

Выполнить команду /bin/ls через три дня в 17 часов

```
[root@openscaler -]# at 5pm+3 days
warning: commands will be executed using /bin/sh
at> /bin/ls
at> <EOT>
job 4 at Fri Dec 18 17:00:00 2020
```

Вывести точное время в указанный файл завтра в 17:20

```
[root@openscaler -]# at 17:20 tomorrow
warning: commands will be executed using /bin/sh
at> date > /root/2020.log
at> <EOT>
job 3 at Wed Dec 16 17:20:00 2020
```

8.1.2 Циклическое исполнение команд (Cron)

Cron - утилита для планирования задач в операционной системе. Она позволяет запускать произвольные скрипты или команды автоматически в определенное время, с определенной периодичностью или при возникновении определенных событий в системе.

Следующие файлы ограничивают доступ к планировщику (по умолчанию они не существуют):

- **/etc/cron.allow** - если существует, пользователи указанные в этом файле имеют возможность запуска заданий планировщика;
- **/etc/cron.deny** - если существует, пользователи указанные в этом файле НЕ имеют возможность запуска заданий планировщика.

Если cron.allow существует, только пользователям, перечисленным в нем, разрешено использовать cron, при этом файл cron.deny игнорируется. Если cron.allow не существует, пользователям, указанным в cron.deny, не разрешается использовать планировщик cron.

Формат записей в обоих файлах - одно имя пользователя в каждой строке. Пробелы не разрешены.

Файлы контроля доступа считываются каждый раз, когда пользователь пытается добавить или удалить задачу cron.

Демон cron проверят конфигурационные файл **/etc/crontab** и файлы расположенные в директории **/etc/cron.*/** и **/var/spool/cron/.**

Для того чтобы пользователю настроить исполнение своих команд по графику необходимо использовать команду crontab для модификации конфигурационных файлов. Ключевые опции использования команды crontab представлены в таблице 23.

Таблица 23. Ключевые опции crontab

Опции запуска команды crontab	Описание
crontab -e	Редактирование или создание файла расписания для текущего пользователя
crontab -l	Вывод содержимого расписания текущего пользователя
crontab -r	Удаление файла расписания текущего пользователя
crontab -u user	Работа с расписаниями конкретных пользователей. Доступно только суперпользователю
:wq!	Как и в редакторе Vl, произведите сохранение измененного файла конфигурации

Каждая запись в crontab состоит из шести полей, указываемых в порядке показанном ниже:

минуты часы дни месяцы день_недели команда

```
*   *   *   *   * команда_для_запуска
-   -   -   -   -
|   |   |   |   |
|   |   |   |   +----- день недели (0 - 6) (Воскресенье = 0 или 7)
|   |   |   +------- месяц (1 - 12)
|   |   +--------- день месяца (1 - 31)
|   +----------- час (0 - 23)
+------------- минуты (0 - 59)
```

Знак ***** означает любое значение, например, если поле минуты содержит *, то задача будет выполняться каждую минуту. Если вы хотите задать конкретное значение для поля, например, каждый день в 3 часа утра, то нужно написать 0 3 * * *.

Использование символа “**,**” позволяет перечислить несколько условий при которых должен отработать скрипт или команда

Символ тире “**-**” задает диапазоны активных значений

Примеры конфигурационных строк:

- Запускать скрипт backupscript.sh каждые 5 минут
  - ***/5 * * * * /root/backupscript.sh**
- Запускать скрипт backupscript.sh в 01.00 каждый день
  - **0 1 * * * /root/backupscript.sh**
- Запускать скрипт backupscript.sh в 03.15 каждый первый день месяца
  - **15 3 1 * * /root/backupscript.sh**

8.2 УПРАВЛЕНИЕ СЕТЕВЫМИ НАСТРОЙКАМИ

8.2.1 Сетевая модель OSI

Сетевая модель OSI (англ. open systems interconnection) - концептуальная модель, которая обобщает и стандартизирует представление средств сетевого взаимодействия в телекоммуникационных и компьютерных системах, независимо от их внутреннего устройства и используемых технологий. Модель OSI была разработана в 1984 году Международной организацией стандартизации (ISO). Основной целью ее создания был поиск решения проблемы несовместимости устройств, использующих различные коммуникационные протоколы, путем перехода на единый, общий для всех систем стек протоколов.

Концепция семиуровневой модели была описана в работе Чарльза Бахмана. Данная модель подразделяет коммуникационную систему на уровни абстракции (англ. "abstraction layers"). В модели OSI средства взаимодействия делятся на семь уровней: прикладной, представления, сеансовый, транспортный, сетевой, канальный и физический. Каждый из которых:

- имеет дело с совершенно определенным аспектом взаимодействия сетевых устройств
- обслуживает уровень, находящийся непосредственно над ним, и обслуживается уровнем, находящимся под ним

Каждый из семи уровней характеризуется типом данных (PDU, сокращение от англ. protocol data units), которым данный уровень оперирует и функционалом, который он предоставляет слою, находящемуся выше него. Предполагается, что пользовательские приложения обращаются только к самому верхнему (прикладному) уровню, однако на практике это выполняется далеко не всегда.

Базовое описание уровней модели представлено в таблице 24.

Таблица 24. Ключевые опции crontab

Уровень	Функции	PDU	Примеры
7. Прикладной	Некоторое высокоуровневое API	Данные	HTTP, FTP
6. Представительский	Представление данных между сетевым сервисом и приложением	Данные	ASCII, EBCDIC, JPEG
5. Сеансовый	Управление сеансами: продолжительный обмен информацией в виде множества передач между нодами	Данные	RPC, PAP
4. Транспортный	Надёжная передача сегментов между двумя нодами в сети	Сегменты/ Датаграммы	TCP, UDP
3. Сетевой	Структуризация и управление множеством нод в сети	Пакеты	Ipv4, IPv6
2. Канальный	Надёжная передача датафреймов между двумя нодами соединённых физическим уровнем	Фреймы	PPP, IEEE 802.2, Ethernet
1. Физический	Передача и приём потока байтов через физическое устройство	Биты	USB, витая пара

Прикладной уровень (Application layer)

Самый верхний уровень модели, предоставляет набор интерфейсов для взаимодействия пользовательских процессов с сетью. Единицу информации, которой оперируют три верхних уровня модели OSI, принято называть сообщение (англ. message).

Прикладной уровень выполняет следующие функции:

- Позволяет приложениям использовать сетевые службы (например удалённый доступ к файлам)
- Идентификация пользователей по их паролям, адресам, электронным подписям
- Предоставление приложениям информации об ошибках
- Определение достаточности имеющихся ресурсов
- Управление данными, которыми обмениваются прикладные процессы и синхронизация взаимодействия прикладных процессов

К числу наиболее распространенных протоколов верхних трех уровней относятся:

- FTP (File Transfer Protocol) протокол передачи файлов
- HTTP (HyperText Transfer Protocol)
- TELNET
- RDP (Remote Desktop Protocol)

Уровень представления (Presentation layer)

Уровень представления занимается представлением данных, передаваемых прикладными процессами в нужной форме. Данные, полученные от приложений с прикладного уровня, на уровне представления преобразуются в формат подходящий для передачи их по сети, а полученные по сети данные преобразуются в формат приложений. Также кроме форматов и представления данных, данный уровень занимается конвертацией структур данных, используемых различными приложениями. Другой функцией, выполняемой на уровне представлений, является шифрование данных, которое применяется в тех случаях, когда необходимо защитить передаваемую информацию от доступа несанкционированными получателями.

Как и прикладной уровень, уровень представления оперирует напрямую сообщениями. Уровень представления выполняет следующие основные функции:

- Генерация запросов на установление/завершение сеансов взаимодействия прикладных процессов
- Согласование представления данных между прикладными процессами
- Конвертация форм представления данных
- Шифрование данных

Примеры протоколов данного уровня:

- AFP — Apple Filing Protocol
- ICA — Independent Computing Architecture
- LPP — Lightweight Presentation Protocol
- NCP — NetWare Core Protocol

Сеансовый уровень (Session layer)

Сеансовый уровень контролирует структуру проведения сеансов связи между пользователями. Он занимается установкой, поддержанием и прерыванием сеансов, фиксирует, какая из сторон является активной в данный момент, осуществляет синхронизацию обмена информацией между пользователями, что также позволяет устанавливать контрольные точки.

На сеансовом уровне определяется, какой будет передача между двумя прикладными процессами:

- полудуплексной (процессы будут передавать и принимать данные по очереди)
- дуплексной (процессы будут передавать данные, и принимать их одновременно)

Как 2 уровня над ним, сеансовый уровень использует сообщения в качестве PDU.

Основные функции:

- Установление и завершение на сеансовом уровне соединения между взаимодействующими приложениями
- Синхронизация сеансовых соединений
- Установление в прикладном процессе меток, позволяющих после отказа либо ошибки восстановить его выполнение от ближайшей метки
- Прекращение сеанса без потери данных
- Передача особых сообщений о ходе проведения сеанса

Примеры протоколов сеансового уровня:

- ADSP (AppleTalk Data Stream)
- ASP (AppleTalk Session)
- RPC (Remote Procedure Call)
- PAP (Password Authentication Protocol)

Транспортный уровень (Transport layer)

Транспортный уровень предназначен для передачи надежной последовательностей данных произвольной длины через коммуникационную сеть от отправителя к получателю. Уровень надежности может варьироваться в зависимости от класса протокола транспортного уровня. Так например UDP гарантирует только целостность данных в рамках одной датаграммы и не исключает возможности потери/дублирования пакета или нарушения порядка получения данных; TCP обеспечивает передачу данных, исключающую потерю данных или нарушение порядка их поступления или дублирования, может перераспределять данные, разбивая большие порции данных на фрагменты и наоборот, склеивая фрагменты в один пакет.

Модель OSI определяет пять классов сервиса, предоставляемых транспортным уровнем. Эти виды сервиса отличаются качеством предоставляемых услуг: срочностью, возможностью восстановления прерванной связи, наличием средств мультиплексирования нескольких соединений между различными прикладными протоколами через общий транспортный протокол, а главное способностью к обнаружению и исправлению ошибок передачи, таких как искажение, потеря и дублирование пакетов. В функции транспортного уровня входят:

- Управление передачей по сети и обеспечение целостности блоков данных
- Обнаружение ошибок, частичная их ликвидация
- Восстановление передачи после отказов и неисправностей
- Разбиение данных на блоки определенного размера
- Предоставление приоритетов при передаче блоков (нормальная или срочная)
- Подтверждение передачи.

Транспортный уровень использует сегменты или датаграммы в качестве основного типа данных.

Примеры протоколов:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- SCTP (Stream Control Transmission Protocol)

Сетевой уровень (Network layer)

Сетевой уровень предоставляет функционал для определения пути передачи пакетов данных между клиентами, подключенными к одной коммуникационной сети. На данном уровне решается проблема маршрутизации (выбора оптимального пути передачи данных), трансляцией логических адресов в физические, отслеживанием неполадок в сети.

В рамках сетевого уровня надежность доставки сообщений не гарантируется; сетевой уровень может реализовывать соответствующий функционал, но не обязан это делать. Роль PDU исполняют пакеты (англ. packet).

Сетевой уровень выполняет функции:

- Обнаружение и исправление ошибок, возникающих при передаче через коммуникационную сеть
- Упорядочение последовательностей пакетов
- Маршрутизация и коммутация
- Сегментирование и объединение пакетов
-

Наиболее часто на сетевом уровне используются протоколы:

- IP/IPv4/IPv6 (Internet Protocol) сетевой протокол стека TCP/IP
- IPX (Internetwork Packet Exchange, протокол межсетевого обмена)
- AppleTalk

Канальный уровень (Data link layer)

Канальный уровень предназначен для передачи данных между двумя узлами, находящихся в одной локальной сети. Роль PDU исполняют фреймы (англ. frame). Фреймы канального уровня не пересекают границ локальной сети, что позволяет данному уровню сосредоточиться на локальной доставке (фактически межсетевой доставкой занимаются более высокие уровни).

Заголовок фрейма формируется из аппаратных адресов отправителя и получателя, что позволяет однозначно определить устройство, которое отправило данный фрейм и устройство, которому он предназначен. При этом никакая часть адреса не может быть использована, чтобы определить некую логическую/физическую группу к которой принадлежит устройство.

Канальный уровень состоит из двух подуровней: LLC и MAC.

Канальный уровень выполняет функции:

- LLC Multiplexing: Интерфейс между сетевым уровнем и MAC, чтобы несколько различных протоколов сетевого уровня могли сосуществовать.
- LLC Flow control: Механизм ограничении скорости передачи данных при медленном приёмнике
- LLC Error control: Определение (и иногда исправление) ошибок с помощью чексумм
- MAC Adressing mechanism: Адрессация на основе уникальных MAC-адресов
- MAC Channel access control mechanism: Предоставляет протокол множественного доступа

Наиболее часто на канальной уровне используются протоколы:

- PPP (Point-To-Point Protocol, протокол прямого соединения между двумя узлами)
- SLIP (Serial Line Internet Protocol, предшественник PPP, который всё ещё используется в микроконтроллерах)
- Ethernet II framing

Физический уровень (Physical layer)

Физический уровень описывает способы передачи потока бит через дата линк, соединяющий сетевые устройства. Поток байт может быть сгруппирован в слова и сконвертирован в физический сигнал, который посылается через некоторое устройство.

Здесь специфицируются такие низкоуровневые параметры как частота, амплитуда и модуляция.

Физический уровень выполняет функции:

Побитовая доставка
- Физическое кодирование (способ представления данных в виде импульсов)
- LLC Error control: Определение (и иногда исправление) ошибок с помощью чексумм
- MAC Adressing mechanism: Адрессация на основе уникальных MAC-адресов
- MAC Channel access control mechanism: Предоставляет протокол множественного доступа

Наиболее часто на физическом уровне используются протоколы:

- Ethernet physical layer (семейство стандартов с оптическими или электрическими свойствами соединений между устройствами)
- USB

8.2.2 Адрес IPv4

IPv4 (Internet Protocol v. 4) — адрес, записанный в 32-битном формате. Имеет вид четырех 8-битных чисел (минимум 0, максимум 255), которые разделены друг от друга точками. Пример: 172.16.255.2.

В общем случае IP-адрес состоит из двух частей (ID-номеров): сети и конкретного узла в ее пределах. Чтобы отличать их в полной записи, используют классы или маски.

8.2.3 Правила именования сетевых интерфейсов

На текущий момент существует две схемы наименования сетевых устройств и интерфейсов в системе. Традиционная и предсказуемая (Predictable).

В традиционной схеме, сетевые карты обычно именуются ethX (где X цифра идентифицирующая устройство в случае если таковых несколько)

В дистрибутиве OpenScaler используется альтернативная, предсказуемая модель именования согласно которой:

- Если устройство интегрировано в материнскую карту и информация о нем поступила в ОС от BIOS или прошивки устройства то используется наименование eno1, eno2 и так далее
- Если карта размещена в слоте PCI-E то наименование меняется на ens1, ens2
- Если системе точно известен интерфейс подключения карты и ее физическое расположение - то наименование меняется на enp2s0 и пр.

Если по сути используется одна сетевая карта или есть желание вернуться к традиционной схеме именования устройств с единым названием, это возможно сделать отредактировав загрузчик операционной системы (GRUB) добавив к строке передаваемых ядру параметров - **net.ifnames=0**

8.2.4 Обзор NetworkManager

NetworkManager — демон управления сетевой подсистемой и набор программ для осуществления настроек сетевых соединений в Linux.

Является стандартом де-факто для большинства дистрибутивов.

На текущий момент обеспечивает возможности настройки:

- Ethernet
- PPPoE
- xDSL
- VPN:
  - PPTP (NetworkManager-pptp)
  - L2TP реализуется при помощи Openswan (NetworkManager-openswan).
  - OpenVPN (NetworkManager-openvpn)
  - VPNC (NetworkManager-vpnc)
  - OpenConnect (NetworkManager-openconnect)
  - WireGuard (NetworkManager-wireguard)
- Беспроводные: Wi-Fi, Bluetooth
- Мобильные: GPRS, 3G и др. — совместно с ModemManager и Mobile-broadband-provider-info.

Для настройки сетевых подключений у NetworkManager есть свой набор инструментария рассчитанного на использование в консоле (CLI интерфейс). Наиболее востребованная утилита это nmcli

nmcli (NetworkManager Command Line Interface) — это утилита командной строки для настройки сети с помощью NetworkManager. Для работы с nmcli используется следующий основной формат:

nmcli [опции] объект { команда | help }

В предыдущей команде вместо “объекта” можно указать один из следующих параметров: **general, networking, radio, connection** и **device**. Вместо опций можно указать необязательные параметры, такие как **-t**, –terse (для обработки скриптов), **-p**, –pretty (для вывода в понятном для человека формате), и **-h** –help.

Для получения дополнительной информации выполните команду nmcli help.

```
$ nmcli help
Usage: nmcli [OPTIONS] OBJECT {COMMAND | help}
```

Из ключевых, наиболее востребованных опций команды стоит отметить следующие:

- **-a** - запрашивать пропущенные пользователем параметры
- **-c** - использовать цвета в выводе данных
- **-h** - вывести справочную информацию
- **-p** - выводить информацию в структурированном виде
- **-s** - разрешить показ паролей
- **-t** - краткий вывод
- **-v** - показать версию программы
- **-w** - задать таймаут на выполнение операций

Объекты:

- **Help** выдаёт справку о командах ncmcli и их использовании.
- **General** возвращает статус NetworkManager и глобальную конфигурацию.
- **Networking** включает команды для запроса состояния сетевого подключения и включения / отключения подключений.
- **Radio** включает команды для запроса состояния подключения к сети WiFi и включения / отключения подключений.
- **Monitor** включает команды для мониторинга активности NetworkManager и наблюдения за изменениями состояния сетевых подключений.
- **Connection** включает команды для управления сетевыми интерфейсами, для добавления новых соединений и удаления существующих.
- **Device** в основном используется для изменения параметров, связанных с устройствами (например, имени интерфейса) или для подключения устройств с использованием существующего соединения.

- **Secret** регистрирует nmcli в качестве «секретного агента» NetworkManager, который прослушивает тайные сообщения. Эта секция используется редко, потому что nmcli при подключении к сетям по дефолту работает именно так.

Основные команды перечислены ниже.

- Чтобы отобразить общее состояние NetworkManager, выполните следующую команду:
  $ nmcli general status
- Чтобы отобразить все подключения, выполните следующую команду:
  $ nmcli connection show
- Чтобы отобразить только текущие активные подключения, добавьте параметр -a или –active следующим образом:
  $ nmcli connection show --active
- Чтобы отобразить устройство, идентифицированное NetworkManager, и состояние его подключения, выполните следующую команду:
  $ nmcli device status
- Например, чтобы запустить или остановить сетевые интерфейсы, выполните команды nmcli от имени пользователя root:
  # nmcli connection up id enp3s0
  # nmcli device disconnect enp3s0

Выполните следующую команду, чтобы подключить NetworkManager к соответствующему сетевому устройству. Попробуйте найти подходящую конфигурацию подключения и активировать ее.

“$IFNAME” - замените на имя интерфейса требуемой сетевой карты:
$nmcli device connect "$IFNAME"

Выполните следующую команду, чтобы отключить NetworkManager от сетевого устройства и предотвратить автоматическую активацию устройства:
$nmcli device disconnect "$IFNAME"

Данные по выбранному устройству/интерфейсу можно посмотреть выполнив команду:

nmcli dev status
DEVICE TYPE STATE CONNECTION
ens3 ethernet disconnected --
ens9 ethernet disconnected --
lo loopback unmanaged --

Если соответствующая конфигурация подключения не существует, NetworkManager создает и активирует файл конфигурации с использованием настроек по умолчанию.

8.2.5 Подключение к сети Ethernet с использованием nmcli

Выполните следующую команду, чтобы отобразить все доступные сетевые подключения:

$ nmcli con show

Настройка динамического IP адреса

Если для предоставления IP адреса сети используется протокол DHCP, выполните следующую команду, чтобы добавить файл конфигурации сети:

nmcli connection add type ethernet con-name connection-name ifname interface-name

Например, чтобы создать файл конфигурации динамического подключения с именем test, выполните следующую команду от имени суперпользователя root:

```
# nmcli connection add type ethernet con-name test ifname enp1s0
Connection 'test' successfully added.
```

NetworkManager устанавливает для connection.autoconnect значение yes и сохраняет параметр в файле **/etc/sysconfig/network-scripts/ifcfg-test**. В файле **/etc/sysconfig/network-scripts/ifcfg-test** для параметра ONBOOT установлено значение **yes**.

Выполните следующую команду от имени суперпользователя root, чтобы активировать сетевое подключение:

```
# nmcli con up test
Connection successfully activated (D-Bus active
path:/org/freedesktop/NetworkManager/ActiveConnection/5)
```

Выполните следующую команду, чтобы проверить состояние подключения устройств:

$ nmcli device status

```
DEVICE      TYPE  STATE         CONNECTION
enp2s0      ethernet  connected  enp4s0
enp1s0      ethernet  connected  test
virbr0      bridge   connected  virbr0
lo      loopback  unmanaged  --
virbr0-nic  tun       unmanaged  --
```

Для того чтобы добавить статическое сетевое подключение IPv4, выполните следующую команду:

nmcli connection add type ethernet con-name connection-name ifname interface-name ip4 address gw4 address

К примеру, для того чтобы создать файл конфигурации статического подключения с именем “static”, выполните следующую команду от имени суперпользователя root:

```
# nmcli con add type ethernet con-name static ifname enp3s0 ip4 192.168.4.10/24 gw4
192.168.4.254
```

Вы также можете указать IPv6-адрес и шлюз для устройства. Ниже приведен пример.

```
# nmcli con add type ethernet con-name static6 ifname enp3s0 ip4 192.168.4.10/24 gw4
192.168.4.254 ip6 abbe::**** gw6 2001:***::*
Connection 'static' successfully added.
```

NetworkManager устанавливает значение **manual** для внутреннего параметра i**pv4.method**, значение yes для параметра **connection.autoconnect** и записывает эту настройку в файл **/etc/sysconfig/network-scripts/ifcfg-static**. В файле для BOOTPROTO установлено значение none, а для ONBOOT — значение yes.

Выполните следующую команду от имени суперпользователя root, чтобы задать IPv4-адреса двух DNS-серверов:

```
# nmcli con mod static ipv4.dns "1.2.3.4 1.2.3.5"
```

Выполните следующую команду от имени суперпользователя root, чтобы задать IPv6-адреса двух DNS-серверов:

```
# nmcli con mod net-static ipv6.dns "2001:4860:4860::**** 2001:4860:4860::****"
```

Выполните следующую команду от имени пользователя root, чтобы активировать сетевое подключение:

```
# nmcli con up net-static ifname enp3s0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/6)
```

Выполните следующую команду, чтобы проверить состояние подключения устройств:

```
$ nmcli device status

DEVICE      TYPE  STATE         CONNECTION
enp2s0      ethernet  connected  enp2s0
enp1s0      ethernet  connected  static
virbr0      bridge connected  virbr0
lo      loopback  unmanaged  --
virbr0-nic  tun       unmanaged  --
```

Выполните следующую команду, чтобы просмотреть сведения о подключении (используйте параметры -p и –pretty, чтобы добавить заголовок и сегмент в выходные данные):

```
$ nmcli -p con show net-static
================================================================================
Connection profile details (net-static )
================================================================================
connection.id:                          net-static
connection.uuid:                        b9f18801-6084-4aee-af28-c8f0598ff5e1
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              enp3s0
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1578988781
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
connection.mdns:                        -1 (default)
connection.llmnr:                       -1 (default)
```

Настройка статического маршрута

Выполните команду nmcli, чтобы настроить статический маршрут для сетевого подключения:

$ nmcli connection modify enp3s0 +ipv4.routes "192.168.122.0/24 10.10.10.1"

Выполните следующую команду, чтобы настроить статический маршрут с помощью редактора:

```
$ nmcli con edit type ethernet con-name enp3s0
===| 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, ipv4,
ipv6, dcb
nmcli> set ipv4.routes 192.168.122.0/24 10.10.10.1
nmcli>
nmcli> save persistent
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation
of the connection.
Do you still want to save? [yes] yes
Connection 'enp3s0' (1464ddb4-102a-4e79-874a-0a42e15cc3c0) successfully saved.
nmcli> quit
```

8.2.6 Конфигурационный файл сетевого интерфейса

Как мы уже ранее описывали, конфигурационные файлы описывающие настроенные варианты подключений NetworkManager'а представлены в каталоге

/etc/sysconfig/network-scripts/<имя подключения>

Ознакомимся с содержимым данных файлов и значениям параметров в них указанными.

К примеру, конфигурационный файл статического подключения ifcfg-enp4s0 выглядит следующим образом:

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
IPADDR=192.168.0.10
PREFIX=24
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp4s0static
UUID=08c3a30e-c5e2-4d7b-831f-26c3cdc29293
DEVICE=enp4s0
ONBOOT=yes
```

Для того чтобы создать вручную конфигурационный файл для интерфейса em1 с динамическими параметрами сети и активацией при загрузке операционной системы - используйте конфигурационный файл из примера выше и измените следующие параметры

DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes

Для того чтобы интерфейс отправлял DHCP альтернативное имя хоста используйте следующую опцию

DHCP_HOSTNAME=hostname

Для того чтобы игнорировать получаемые от DHCP сервера данные о DNS и предотвратить затирание файла /etc/resolv.conf используйте опцию

PEERDNS=no

Для того чтобы самостоятельно вручную указать адреса требуемых DNS серверов используйте опции

DNS1=ip-address
DNS2=ip-address

ip-address — это IP-адрес DNS-сервера. Это позволяет сетевому сервису обновлять файл /etc/resolv.conf, используя указанный DNS-сервер.

8.2.7 Настройка имени хоста (hostname)

Существует три типа имен хостов: **static**, **transient** и **pretty**.

- **static**: статическое имя хоста, которое может быть задано пользователями и сохранено в файле /etc/hostname.
- **transient**: динамическое имя хоста, поддерживаемое ядром. Начальное значение — статическое имя хоста. Значение по умолчанию — localhost. Это значение можно изменить, когда работает сервер mDNS или DHCP.
- **pretty**: гибкое имя хоста, которое можно задавать в любой форме (включая специальные символы и пробелы). На статические и динамические имена хостов распространяются общие ограничения для доменных имен.

Важно помнить, что статические и динамические имена хостов могут содержать только буквы (a–z и A–Z), цифры (0–9), дефисы (-), символы подчеркивания (_) и точки (.). Имена хостов не могут начинаться или заканчиваться точкой (.), а также содержать две последовательные точки (.). Имя хоста может содержать не более 64 символов.

8.2.7.1 Настройка имени хоста (hostnamectl)

Выполните следующую команду, чтобы просмотреть текущее имя хоста:

$ hostnamectl status

Выполните следующую команду от имени суперпользователя root, чтобы задать все имена хостов:

 # hostnamectl set-hostname name

Выполните следующую команду от имени суперпользователя root, чтобы задать конкретное имя хоста:

 # hostnamectl set-hostname name [option...]

В качестве option можно указать один или несколько из параметров –pretty, –static и –transient

Если –static или –transient используется вместе с –pretty, имена хостов типа static или transient будут упрощены до имен хостов типа pretty с заменой пробелов дефисами (-) и удалением специальных символов.

При задании имени хоста типа pretty используйте кавычки, если имя хоста содержит пробелы или одиночные кавычки. Ниже приведен пример.

 # hostnamectl set-hostname "Stephen's notebook" --pretty

Чтобы выполнить сброс конкретное имя хоста и восстановить его в формате по умолчанию, выполните следующую команду от имени суперпользователя root:

 # hostnamectl set-hostname "" [option...]

В предыдущей команде “” — это пустая строка символов, а в качестве option можно указать один или несколько из параметров –pretty, –static и –transient.

Чтобы изменить имя хоста в удаленной системе, выполните команду hostnamectl с параметром -H или –host от имени суперпользователя root.

 # hostnamectl set-hostname -H [username]@hostname new_hostname

В предыдущей команде hostname обозначает имя настраиваемого удаленного хоста, username — определенное пользователем имя, а new_hostname — новое имя хоста. Команда hostnamectl используется для подключения к удаленной системе по протоколу SSH.

8.2.7.2 Настройка имени хоста (nmcli)

Чтобы запросить статическое имя хоста, выполните следующую команду:

$ nmcli general hostname

Чтобы присвоить статическому хосту имя host-server, выполните следующую команду от имени суперпользователя root:

 # nmcli general hostname host-server

Чтобы система могла обнаружить изменение статического имени хоста, выполните следующую команду от имени пользователя root для перезагрузки сервиса hostnamed:

 # systemctl restart systemd-hostnamed

8.3 УПРАВЛЕНИЕ ПРОЦЕССАМИ

Говоря в общих чертах, программа есть статический файл с набором исполняемых инструкций (кода). Процесс - есть сущность программы в момент ее исполнения.
Процесс - абстракция, характеризующая программу во время выполнения.
Понятие процесса характеризует некоторую совокупность набора исполняющихся команд, ассоциированных с ним ресурсов и текущего момента его выполнения, находящуюся под управлением операционной системы.

8.3.1 Классификация процессов в ОС

В системе все существующие процессы можно разбить на две большие группы - системные и пользовательские процессы.

Системные процессы не имеют собственных исполняемых файлов и запускаются из ядра операционной системы. Системные процессы стартуют при загрузке операционной системы и выполняются в течение всего сеанса работы. Они запускаются от имени операционной системы и обладают соответствующими правами. Эти процессы не общаются с пользователем, и он может их обнаружить только просматривая список процессов. Соответственно, пользователь, если он не является администратором системы, не имеет права распоряжаться этими процессами, посылая им сигналы. Есть один процесс, который по своей сути является системным, но запускается не из ядра, а из отдельного исполняемого файла с именем init. Этот процесс является прародителем всех остальных процессов.

Пользовательские процессы запускаются из исполняемого (бинарного) файла пользователем. Они могут выполняться в интерактивном или фоновом режимах и срок их выполнения ограничен продолжительностью сеанса работы пользователя. Пользовательские процессы наследуют идентификатор пользователя и, как правило, имеют соответствующие права на доступ к объектам. Самый важный пользовательский процесс - командный интерпретатор, обеспечивающий диалоговый режим с пользователем. К моменту, когда система предоставит пользователю право управлять собой, в ней уже будет существовать несколько десятков системных и сервисных процессов. Система изолирует пользовательские процессы друг от друга и от системных процессов. Каждый процесс выполняется в своем виртуальном адресном пространстве и других процессов «не видит». В многозадачной операционной системе пользовательские процессы не имеют права читать данные чужого процесса, записывать свои данные в его адресное пространство, отбирать у других задач вычислительное время и доступ к устройствам ввода-вывода.

8.3.2 Состояния процессов в ОС

Процессы в системе обычно могут находится в одном из пяти состояний: новый (рожденный), готовый к исполнению, ожидающий, исполняющийся и закончивший исполнение. График перехода процесса из состояния в состояние представлен на рисунке 22.

Рисунок 22. Состояния процесса в операционной системе

- **Новый (рожденный)** Новый, появляющийся процесс в системе. Это первое исходное состояние процесса. Родившийся процесс переводится в состояние готовность. При рождении процесс получает в свое распоряжение:
  - адресное пространство, в которое загружается программный код процесса;
  - выделяются стек и системные ресурсы;
  - устанавливается начальное значение программного счетчика этого процесса
- **Исполнение процесса** Операционная система выбирает один процесс для последующего исполнения. Для избранного процесса операционная система обеспечивает наличие в оперативной памяти информации, необходимой для его дальнейшего выполнения. Состояние процесса изменяется на исполнение. Восстанавливаются значения регистров для данного процесса. Управление передается команде, на которую указывает счетчик команд процесса. Все данные, необходимые для восстановления контекста, извлекаются из PCB процесса, над которым совершается операция.
- **Приостановка процесса (ожидание)** Работа процесса, находящегося в состоянии исполнение, приостанавливается в результате какого-либо прерывания.Процессор автоматически сохраняет счетчик команд и, возможно, один или несколько регистров в стеке исполняемого процесса. Передает управление по специальному адресу обработки данного прерывания.
- **Приостановка процесса (блокировка)** Процесс блокируется, когда он не может продолжать работу, не дождавшись возникновения какого-либо события в вычислительной системе. Процесс обращается к операционной системе с помощью определенного системного вызова. Операционная система обрабатывает системный вызов и, при необходимости сохранив нужную часть контекста процесса в его PCB, переводит процесс из состояния исполнение в состояние ожидание.
- **Разблокировка процесса** После возникновения в системе какого-либо события операционной системе нужно точно определить, какое именно событие произошло. Затем операционная система проверяет, находился ли некоторый процесс в состоянии ожидание для данного события, и если находился, переводит его в состояние готовность, выполняя необходимые действия, связанные с наступлением события.
- **Завершение исполнения (Зомби)** Процесс, который завершил свое выполнение и был прерван с помощью системного вызова, но все еще имеет свою запись в таблице процессов системы чтобы дать родительскому процессу считать код завершения.

8.3.3 ID процессов и процессы потомки

Каждая программа запущенная в ОС может быть представлена не одним а множеством процессов и подпроцессов. Чтобы однозначно идентифицировать каждый процесс в системе, им присваиваются Process ID (PID). У процессов также имеется PPID (Parent PID) определяющий процесс-родителя. Все процессы в системе, даже те что не имеют своих потомков, являются потомками процесса PID 1 (init).

С исчезновением процесса родителя исчезают и процессы потомки, но не наоборот.

8.3.4 Просмотр процессов в ОС

Команда ps

Команда ps (process status) позволяет получить исчерпывающую информацию по всем присутствующим в системе процессам.

Рассмотрим опции утилиты. Они делятся на два типа - те, которые идут с дефисом (Unix стиль) и те, которые используются без дефиса (BSD стиль). Лучше пользоваться только опциями Unix, но мы рассмотрим и одни и другие. Заметьте, что при использовании опций BSD, вывод утилиты будет организован в BSD стиле.

Среди ключевых опций использования команды стоит отметить:

- **A, -e, (a)** - выбрать все процессы;
- **-a** - выбрать все процессы, кроме фоновых;
- **-d, (g**) - выбрать все процессы, даже фоновые, кроме процессов сессий;
- **-N** - выбрать все процессы кроме указанных;
- **-C** - выбирать процессы по имени команды;
- **-G** - выбрать процессы по ID группы;
- **-p, (p)** - выбрать процессы PID;
- **--ppid** - выбрать процессы по PID родительского процесса;
- **-s** - выбрать процессы по ID сессии;
- **-t, (t)** - выбрать процессы по tty;
- **-u, (U)** - выбрать процессы пользователя.

Опции форматирования:

- **-c** - отображать информацию планировщика;
- **-f** - вывести максимум доступных данных, например, количество потоков;
- **-F** - аналогично -f, только выводит ещё больше данных;
- **-l** - длинный формат вывода;
- **-j, (j)** - вывести процессы в стиле Jobs, минимум информации;
- **-M, (Z)** - добавить информацию о безопасности;
- **-o, (o)** - позволяет определить свой формат вывода;
- **--sort, (k)** - выполнять сортировку по указанной колонке;
- **-L, (H)**- отображать потоки процессов в колонках LWP и NLWP;
- **-m, (m)** - вывести потоки после процесса;
- **-V, (V)** - вывести информацию о версии;
- **-H** - отображать дерево процессов.

В зависимости от выбранных опций вывод команды может включать следующие поля описания процессов:

- **UID** - пользователь, от имени которого запущен процесс;
- **PID** - идентификатор процесса;
- **PPID** - идентификатор родительского процесса;
- **C** - процент времени CPU, используемого процессом;
- **STIME** - время запуска процесса;
- **TTY** - терминал, из которого запущен процесс;
- **TIME** - общее время процессора, затраченное на выполнение процессора;
- **CMD** - команда запуска процессора;
- **LWP** - показывает потоки процессора;
- **PRI** - приоритет процесса.
- **SZ** - это размер процесса в памяти;
- **RSS** - реальный размер процесса в памяти;
- **PSR** - ядро процессора, на котором выполняется процесс.
- **STAT** - текущий статус процесса:
- **D** - неприрываемый
- **R** - исполняемый
- **S** - остановленный по прерыванию
- **T** - заблокированный (остановленный, ждет события)
- **Z** - закончил исполнение (зомби)

pstree

Команда отображает все процессы системы в виде древовидной структуры.

Среди ключевых опций использования команды стоит отметить:

- **-a** - показывает команды с аргументами в командной строке. Если командная строка процесса is swapped out, то процесс показывается заключенным в круглые скобки. -a неявно запрещает компактную форму вывода.
- **-c** Запрещает компактную форму вывода одинаковых поддеревьев . По умолчанию, одинаковые поддеревья объединяются всякий раз когда возможно.
- **-G** - Использовать для вывода дерева символы соответствующие VT100.
- **-h** -Подсвечивает текущий процесс и его предков.
- **-l** - Показывает длинные строки. По умолчанию, строки обрезаются до ширины дисплея или до 132 если вывод посылается на не-tty устройство или если ширина дисплея неизвестна.
- **-n** - Сортирует процессы с одинаковым предком по идентификатору процесса (PID) вместо сортировки по имени. (Числовая сортировка.)

- **-p** - Показывает идентификаторы процессов PIDs.Идентификаторы процессов (PIDs) показываются десятиричным числом , заключенным в курглые скобки после каждого имени процесса. -p запрещает вывод в компактной форме.
- **-u** - Показывает uid . Всякий раз когда uid процесса отличается от uid родителя , то новый uid показывается после имени процесса ,заключенным в круглые скобки.
- **-U** - использует UTF-8 (Unicode) символы unicode для рисования дерева.
- **-V** - Показывает информацию о версии программы.

top

Команда позволяет в режиме реального времени отслеживать и управлять процессами в операционной системе.

Среди ключевых опций использования команды стоит отметить:

- **-v** - вывести версию программы;
- **-b** - режим только для вывода данных, программа не воспринимает интерактивных команд и выполняется пока не будет завершена вручную;
- **-c** - отображать полный путь к исполняемым файлам команд;
- **-d** - интервал обновления информации;
- **-H** - включает вывод потоков процессов;
- **-i** - не отображать процессы, которые не используют ресурсы процессора;
- **-n** - количество циклов обновления данных, после которых надо закрыть программу;
- **-o** - поле, по которому надо выполнять сортировку;
- **-O** - вывести все доступные поля для сортировки;
- **-p** - отслеживать только указанные по PID процессы, можно указать несколько PID;
- **-u** - выводить только процессы, запущенные от имени указанного пользователя.

В режиме интерактивной работы с приложением доступны также следующие горячие клавиши:

- **h** - вывод справки по утилите;
- **q** или Esc - выход из top;
- **A** - выбор цветовой схемы;
- **d** или **s** - изменить интервал обновления информации;
- **H** - выводить потоки процессов;
- **k** - послать сигнал завершения процессу;
- **W** - записать текущие настройки программы в конфигурационный файл;
- **Y** - посмотреть дополнительные сведения о процессе, открытые файлы, порты, логи и т. д;
- **Z** - изменить цветовую схему;
- **l** - скрыть или вывести информацию о средней нагрузке на систему;

- **m** - выключить или переключить режим отображения информации о памяти;
- **x** - выделять жирным колонку, по которой выполняется сортировка;
- **y** - выделять жирным процессы, которые выполняются в данный момент;
- **z** - переключение между цветным и одноцветным режимами;
- **c** - переключение режима вывода команды, доступен полный путь и только команда;
- **F** - настройка полей с информацией о процессах;
- **o** - фильтрация процессов по произвольному условию;
- **u** - фильтрация процессов по имени пользователя;
- **V** - отображение процессов в виде дерева;
- **i** - переключение режима отображения процессов, которые сейчас не используют ресурсы процессора;
- **n** - максимальное количество процессов, для отображения в программе;
- **L** - поиск по слову;
- **<>** - перемещение поля сортировки вправо и влево;

Пример внешнего вида интерфейса программы представлен на рисунке 23

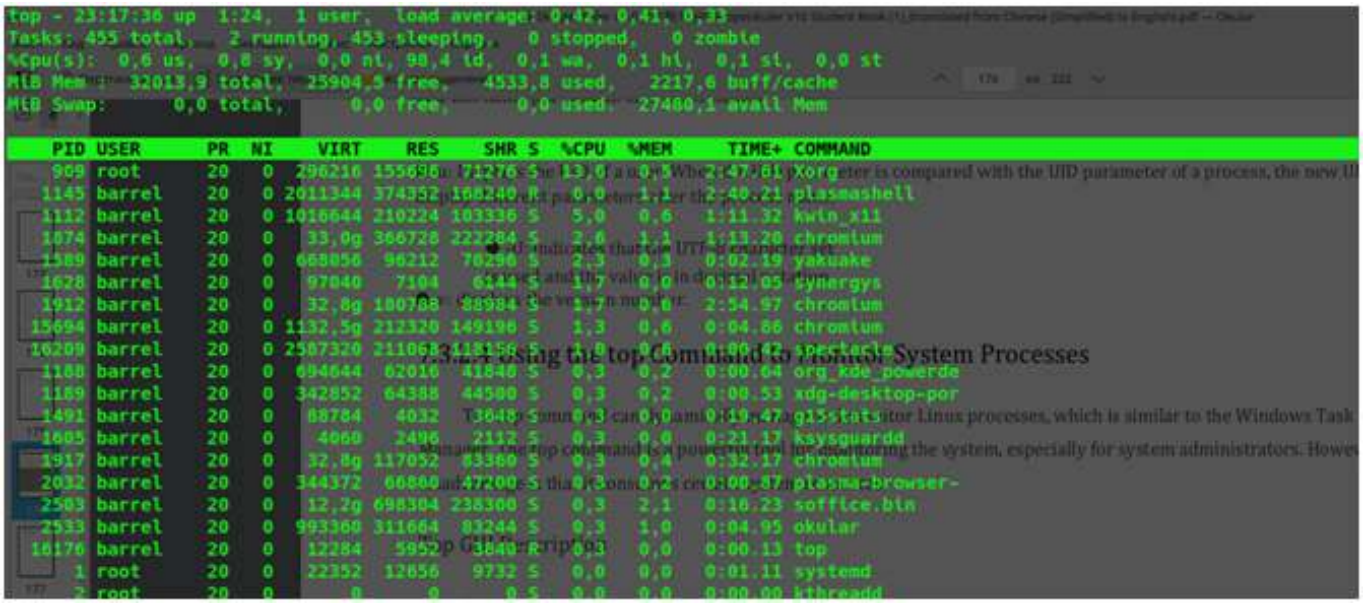

Рисунок 23. Интерфейс программы top

Интерфейс можно разделить на две рабочие зоны:

- Верхняя рабочая зона содержит сведения о времени работы сервера, свободных и занятых ресурсах, пользователях;
- Основная рабочая зона - это динамически обновляемая таблица, содержащая сведения о процессах.

В верхнем левом углу экрана отображено текущее время - 23:17:36, за которым следует время безотказной работы системы 1:24. Далее в строке идет количество активных сеансов пользователя.

В разделе **Tasks** отображается статистика процессов, выполняемых в вашей системе. Далее перечислено общее количество процессов, активные, спящие, остановленные и процессы-зомби.

Раздел использования **CPU** показывает процентное время процессорного времени, затрачиваемого на различные задачи.

Последние 2 строки показывают информацию об использовании памяти в системе. Строки **Mem** и **Swap** отображают информацию о ОЗУ и области подкачки соответственно. Указаны значения общего, свободного, используемого объема и кеша. **Avail Mem** - это объем памяти, который может быть выделен для процессов, не использую большую область диска.

Значение всех столбцов представлено в таблице 25.

Таблица 25. Значение столбцов интерфейса команды top

Столбец	Значение
PID	Это идентификатор процесса, уникальное положительное целое число, которое идентифицирует процесс.
USER	Это эффективное имя пользователя (соответствующее идентификатору пользователя) пользователя, который запустил этот процесс. Linux назначает реальный идентификатор пользователя и эффективный идентификатор пользователя для процессов; последний позволяет процессу действовать от имени другого пользователя. Например, пользователь, не являющийся пользователем root, может с правами root установить пакет.
PR	Поле показывает приоритет выполнения процесса с точки зрения ядра.
NI	Поле показывает nice-значение процесса.
VIRT	Общий объем памяти, потребляемый процессом. Он включает в себя код программы, данные, хранящиеся в памяти, а также любые области памяти, которые были выделены по запросу процесса.
RES	Количество памяти, потребляемое процессом в оперативной памяти.
SHR	Объем памяти, совместно используемый другими процессами.
S	В этом поле отображается состояние процесса в однобуквенной форме (R - Runnable, D - Interruptible sleep, S - Uninterruptible sleep, T - Stopped, Z - Zombie).

%CPU	Параметр выражает объем в процентах от общей доступной оперативной памяти ОЗУ.
%MEM	Параметр выражает значение RES в процентах от общей доступной оперативной памяти.
TIME+	Общее время процессора, используемое процессом с момента его начала, с точностью до сотых долей секунды.
COMMAND	Здесь отображено название процессов.

8.3.5 Остановка процессов в ОС (kill)

Команда позволяет принудительно остановить процесс в системе. Базовый синтаксис команды представлен ниже.

kill [опции] [PID]

Среди ключевых опций использования команды стоит отметить:

- **-l** - вывести список всех возможных сигналов и их номеров которые можно передать процессу
- **-p** - только вывести ID процессов не посылая никаких сигналов
- **-s** - указать отправляемый сигнал
- **-u** - указать пользователя

Некоторые из наиболее часто используемых сигналов:

1 HUP (hang up) — повесить.
2 INT (interrupt) — прерывание.
3 QUIT (quit) — выход.
6 ABRT (abort) — прерывания.
9 KILL (non-catchable, non-ignorable kill)
14 ALRM (alarm clock) — будильник.
15 TERM (software termination signal) — Программное обеспечение для прекращения сигнала.

killall

Данная команда позволяет прекратить процесс и всех его потомков. к примеру, завершить исполнение всех процессов относящихся к svnserver

killall svnserver

Чтобы принудительно завершить процесс и всех его потомков добавьте опцию -9 (что будет равносильно команде kill).

8.4 ЛОКАЛИЗАЦИЯ ОС

8.4.1 Установка системной локализации (locale)

Локали (locale) определяют язык, который использует система, а также региональные особенности, такие как денежные знаки, формат чисел, даты и времени и наборы символов. Они используются glibc и некоторыми другими программами или системными библиотеками для рендеринга текста.

Настройки локали хранятся в файле **/etc/locale.conf** и могут быть изменены командой **localectl**. Эти настройки считываются демоном systemd при загрузке системы.

Для того чтобы отобразить текущее состояние языковых настроек, выполните следующую команду:

```
$ localectl status
Пример вывода команды:
# localectl status
 System Locale: LANG=ru_RU.UTF-8
 VC Keymap: us
X11 Layout: us,ru
X11 Variant: ,
X11 Options: grp:alt_shift_toggle
```

Для того чтобы отобразить все доступные на текущий момент локали в системе, выполните следующую команду:

```
$ localectl list-locales
```

Вы можете также применить выборку с использованием команды grep, перечислив, к примеру, только все варианты локалей реализующих поддержку русского языка с помощью следующей команды:

```
localectl list-locales | grep ru
ru_RU.UTF-8
ru_UA.UTF-8
```

Для того чтобы настроить выбранную локаль, необходимо выполнить от имени суперпользователя root следующую команду. В этой команде **locale** указывает выбранную локаль, которую необходимо применить в системе. Запустите команду **localectl list-locales**, чтобы получить список локалей и их кодировок которые могут быть использованы в системе. Измените значение необходимым образом.

```
# localectl set-locale LANG=locale
```

Например, чтобы использовать “Русский язык в кодировке UTF-8”, выполните от имени суперпользователя root следующую команду:

 # localectl set-locale LANG=ru_RU.UTF-8

Внеся вышеописанные изменения, необходимо реавторизоваться в системе (logout/login) или выполнить команду **source /etc/locale.conf** от имени суперпользователя root, чтобы обновить файл конфигурации, и тогда изменения вступят в силу.

8.4.2 Управление раскладками клавиатуры

Настройки раскладки клавиатуры хранятся в файле **/etc/vconsole.conf** и могут быть изменены командой **localectl**. Эти настройки считываются на раннем этапе загрузки демоном systemd.

Для того чтобы отобразить текущие настройки раскладки клавиатуры, выполните следующую команду:

$ localectl status

Пример вывода команды:

```
# localectl status
System Locale: LANG=ru_RU.UTF-8
VC Keymap: us
X11 Layout: us,ru
X11 Variant: ,
X11 Options: grp:alt_shift_toggle
```

Для того чтобы получить список всех доступных в системе раскладок клавиатуры, выполните следующую команду:

$ localectl list-keymaps

Например, команда вывода раскладок с поддержкой русского языка выглядит следующим образом:

```
# localectl list-keymaps | grep ru
cz-rus
dvorak-ru
ge-ru
ng-yoruba
ru
ru-cp1251
....
```

Для того чтобы применить выбранную раскладку клавиатуры, выполните следующую команду от имени суперпользователя root. В этой команде ru указывает раскладку клавиатуры, которую необходимо установить. Чтобы получить диапазон возможных значений, выполните команду localectl list-keymaps. Измените значение необходимым образом.

```
$ localectl set-keymap ru
```

Раскладка клавиатуры будет аналогичным образом применяться к графическим пользовательским интерфейсам в случае их использования пользователем.

После проведения настройки можно осуществить проверку, применена ли установка:

```
# localectl status
System Locale: LANG=ru_RU.UTF-8
VC Keymap: ru
X11 Layout: us,ru
X11 Variant: ,
X11 Options: grp:alt_shift_toggle
```

8.4.3 Установка даты и времени

В этом разделе описано задание системной даты, времени и часового пояса с помощью команд **timedatectl**, **date** и **hwclock**.

timedatectl

Для того чтобы отобразить текущую дату и время, выполните следующую команду:

```
$ timedatectl
```

Пример вывода команды:

```
# timedatectl
               Local time: Пт 2023-04-28 10:33:01 MSK
           Universal time: Пт 2023-04-28 07:33:01 UTC
                 RTC time: Пт 2023-04-28 07:33:01
                Time zone: Europe/Moscow (MSK, +0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Системные часы могут автоматически синхронизироваться с удаленным сервером, используя протокол NTP (Network Time Protocol — протокол сетевого времени). Для включения или отключения NTP выполните от имени суперпользователя root следующую команду. Логическое значение (boolean) равно yes (да) или no (нет), что характеризует включение или отключение NTP для автоматической синхронизации системных часов. Измените значение необходимым образом.

Если на удаленном NTP-сервере включена автоматическая синхронизация системных часов, вы не сможете изменить дату и время вручную. Если вам необходимо вручную изменить дату или время, убедитесь, что автоматическая синхронизация системных часов с помощью NTP отключена. Отключить службу NTP можно командой **timedatectl set-ntp no**.

 # timedatectl set-ntp boolean

К примеру, для того чтобы включить автоматическую удаленную синхронизацию времени, выполните следующую команду:

 # timedatectl set-ntp yes

Для того чтобы изменить текущую дату выполните от имени суперпользователя root следующую команду. В этой команде YYYY указывает год, MM — месяц, а DD — день. Измените эти значения необходимым образом.

 # timedatectl set-time YYYY-MM-DD

К примеру, для того чтобы изменить текущую дату на 19 ноября 2023 г., от имени суперпользователя root выполните следующую команду:

 # timedatectl set-time '2023-11-19'

Перед изменением даты убедитесь, что автоматическая синхронизация системных часов с помощью NTP отключена.

Для того чтобы изменить текущее время, выполните от имени суперпользователя root следующую команду. В этой команде HH указывает час, MM — минуты, а SS — секунды. Измените эти значения необходимым образом.

 # timedatectl set-time HH:MM:SS

К примеру, для того чтобы изменить текущее время на 14:38:14, выполните следующую команду:

 # timedatectl set-time 14:38:14

Также как и перед изменением даты, необходимо убедиться что автоматическая синхронизация системных часов с помощью NTP отключена.

Для того чтобы вывести список всех доступных в системе часовых поясов, выполните следующую команду:

$ timedatectl list-timezones

Для того чтобы изменить текущий часовой пояс, выполните от имени суперпользователя root следующую команду. В этой команде time_zone указывает устанавливаемый часовой пояс. Измените значение необходимым образом.

 # timedatectl set-timezone time_zone

К примеру, вы находитесь в Европе и хотите определить, какой часовой пояс к вам ближе всего. Для этого можно осуществить выборку из полного списка и вывести только список всех доступных часовых поясов в Европе, используя следующую команду:

```
# timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Andorra
Europe/Astrakhan
Europe/Athens
.....
```

Для того чтобы изменить часовой пояс на Europe/Moscow (Европа/Москва), выполните следующую команду:

 # timedatectl set-timezone Europe/Moscow

date

Для того чтобы отобразить текущую дату и время, выполните следующую команду:

$ date

По умолчанию команда date отображает местное время. Чтобы отобразить время в формате всемирного координированного времени (UTC), запустите команду с параметром командной строки **–utc** или **-u**:

$ date --utc

Вы также можете настроить формат отображаемых сведений, указав в командной строке параметр +“format”:

$ date +"format"

В таблице 26 представлены параметры форматирования для команды date

Таблица 26. Параметры форматирования

Параметр формата	Описание
%H	Час в формате HH (например, 17).
%M	Минуты в формате MM (например, 37).
%S	Секунды в формате SS (например, 25).
%d	День месяца в формате DD (например, 15).
%m	Месяц в формате MM (например, 07).
%Y	Год в формате YYYY (например, 2023).
%Z	Аббревиатура часового пояса (например, CEST).
%F	Полная дата в формате YYYY-MM-DD (например, 2023-04-28). Этот параметр эквивалентен %Y-%m-%d.
%T	Полное время в формате HH:MM:SS (например, 18:30:25). Этот параметр эквивалентен %H:%M:%S.

Примеры исполнения команды:

- Отображение текущей даты и времени:
  # date
  Пт 28 апр 2023 10:41:32 MSK

- Отображение текущей даты и времени в формате UTC:
  # date --utc
  Пт 28 апр 2023 07:44:07 UTC

- Настройка вывода команды date:
  $ date +"%Y-%m-%d %H:%M"
  2023-04-28 17:24

Для того чтобы изменить текущее время, запустите команду date с параметром **–set** или **-s** от имени суперпользователя root. В этой команде HH указывает час, MM — минуты, а SS — секунды. Измените эти значения необходимым образом.

 # date --set HH:MM:SS

По умолчанию команда date устанавливает местное время. Чтобы вместо этого установить системные часы в формате UTC, запустите команду с параметром командной строки **–utc** или **-u**:

 # date --set HH:MM:SS --utc

К примеру, для того чтобы изменить текущее время на 13:11:00, выполните от имени суперпользователя root следующую команду:

```
# date --set 13:11:00
```

Для того чтобы изменить текущую дату, запустите команду с параметром командной строки **–set** или **-s** от имени суперпользователя root. В этой команде YYYY указывает год, MM — месяц, а DD — день. Измените эти значения необходимым образом.

```
# date --set YYYY-MM-DD
```

Например, чтобы изменить текущую дату на 19 ноября 2023 г., от имени пользователя root выполните следующую команду:

```
# date --set 2023-11-19
```

hwclock

Команда hwclock позволяет установить часы реального времени (RTC).

Операционная система Linux выделяет следующие типы часов.

- Системные часы: часы текущего ядра Linux.
- Аппаратные RTC-часы: аппаратные часы материнской платы с питанием от батареи. Эти часы можно установить в BIOS с помощью параметра Standard BIOS Feature (Стандартный компонент BIOS).

При запуске операционная система считывает RTC и устанавливает время системных часов на основе времени RTC.

Для того чтобы отобразить текущую дату и время RTC, выполните следующую команду от имени суперпользователя root:

```
# hwclock
```

Пример вывода команды:

```
# hwclock
2023-04-28 11:31:13.609433+03:00
```

Для того чтобы изменить дату и время на текущей операционной системе, запустите от имени суперпользователя root следующую команду. В этой команде dd указывает день, mm — месяц, yyyy — год, HH — час, а MM — минуты. Измените эти значения необходимым образом.

 # hwclock --set --date "dd mm yyyy HH:MM"

Например, чтобы изменить текущее время на 14:55 и дату на 19 ноября 2023г., выполните следующую команду:

 # hwclock --set --date "19 Nov 2023 14:55" --utc

8.5 Вопросы для самопроверки

- Измените имя хоста на OpenScaler
- Создайте автоматическое задание на экспорт текущего времени в файл в домашнем каталоге пользователя каждый день в 17.30
- Запланируйте разовую операцию по проведения копирования всех файлов из каталога /etc в /backup на 02.00 через 4 дня.

9. ПРИЛОЖЕНИЯ И ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

9.1 ФУНКЦИОНАЛЬНО-ТЕХНИЧЕСКИЕ ПРЕИМУЩЕСТВА OPENSCALER

9.1.1 Kernel-Live Update

Технология Kernel Live Upgrade позволяет произвести перезапуск системы Linux без фактического перезапуска сервера.

Поддерживаются архитектуры x86_64 и aarch64.

То есть происходит штатный останов всех служб, а затем вместо полного цикла перезапуска производится запуск новой/другой версии ядра Linux. Данный механизм позволяет сократить время ожидания перезапуска для ВМ примерно 10 сек, для физического сервера 2 мин.

Установка производится командой:

dnf install nvwa

Помимо сокращения времени перезагрузки, демон nvwa может сохранить и состояние процесса, указанного в конфигурационном файле /etc/nvwa/nvwa-restore.yaml. Для этого используется инструмент criu, но это не всегда происходит успешно. Рекомендуется производить ручное сохранение и восстановление состояния нужных процессов, с использованием criu, а не полагаться на утилиту nvwa, а саму nvwa использовать только для смены ядра.

Для смены ядра необходимо передать версию ядра, из каталога **/boot**.

Например для следующих ядер:

```
[root@klu-x86 ~]# ll /boot/vmlinuz*
-rwxr-xr-x. 1 root root 10979584 апр 26 09:50 /boot/vmlinuz-0-rescue-
ca260fe8129848fd88b15e21f15f072b
-rwxr-xr-x. 1 root root 10979584 мар  1 18:21 /boot/vmlinuz-5.10.0-
136.12.0.86.os2203sp1.x86_64
-rwxr-xr-x. 1 root root 10994720 мар 20 11:53 /boot/vmlinuz-5.10.0-
136.22.0.98.os2203sp1.x86_64
```

Команда будет:

```
nvwa update 5.10.0-136.22.0.98.os2203sp1.x86_64
```

или для старого ядра

```
nvwa update 5.10.0-136.12.0.86.os2203sp1.x86_64
```

Таким образом технология реально может быть полезна.

9.1.2 NFS Multipathing

Сетевая файловая система (NFS) – это протокол распределенной файловой системы, первоначально разработанный Sun Microsystems (Sun) в 1984 году. Этот протокол позволяет пользователям NFS-клиентов получать доступ к файлам на NFS-серверах по компьютерным сетям. Поскольку сервис NFS широко используется в таких отраслях, как финансы, EDA, искусственный интеллект и контейнеризация, к производительности и надежности NFS предъявляются более высокие требования.

Традиционная NFS обладает следующими недостатками:

- Доступ к точке монтирования на клиентском хосте можно получить только с помощью одного IP-адреса клиента и одного IP-адреса сервера. Когда между клиентом и сервером существует несколько физических соединений, производительность этих соединений не может быть использована в полной мере.

- Если канал связи точки монтирования стал неисправен, то, даже при условии что у нас есть ещё один канал связи с NFS сервером, переход на этот другой канал не может быть выполнен. В результате на NFS-клиенте возникают сбои при попытке обращения к смонтированном ресурсу NFS.

NFS multipathing разработана для устранения предыдущих дефектов, возникающих при использовании традиционной NFS. Предлагается установить несколько каналов связи между клиентом и сервером в рамках одной точки монтирования для поддержки передачи данных ввода-вывода по нескольким каналам, повышая производительность одной точки монтирования. Кроме того, периодически проверяется состояние соединения, чтобы обеспечить быструю отработку отказа ввода-вывода при сбое соединения.

NFS multipathing предоставляет следующие функции:

- NFSv3 поддерживает алгоритм циклического выбора каналов связи, чтобы сбалансировать их производительность.
- NFSv3 и NFSv4 поддерживают быструю отработку отказа соединения, повышая надежность NFS.
- Предоставляет интерфейс для регистрации алгоритмов выбора пути соединения и позволяет разработчикам сервера NFS настраивать алгоритмы выбора пути.
- Поддерживает периодическое определение доступности канала.
- Позволяет просматривать статус ссылки.

Кто-то может подумать, зачем нужна эта технология, если с 2010 года уже существует расширение протокола pNFS, которое было добавлено в версии NFSv4.1? Но тут есть нюанс в том, что клиентская часть pNFS в Linux реализована на достаточно приемлемом уровне, а вот реализации серверов требуют ещё некоторое количество доработок. Те кто пытался настроить pNFS в Linux сталкивался с рядом ограничений, а именно – некоторые реализации pNFS не поддерживают нужные режимы работы (Block Volume, Flexible Files, OSD2 Objects, Block SCSI, NFSv4.1 Files), или часть их функционала реализована ещё не полностью.

К тому же, следует заметить, что pNFS работает со слоями (layout) и имеет разделение серверов на хранилище метаданных (MDS) и хранилища данных (DS), а представленная в этой статье технология является расширением над классическим NFS v4 с файловым доступом к серверу NFS, у которого имеется несколько сетевых подключений, которые обеспечивают доступность с NFS клиента по множеству маршрутов.

Установка

Данная технология представлена в версии OpenScaler 23.03, которая не является LTS версией и предназначена больше для демонстрации новых технологий, которые в дальнейшем получат своё развитие.

Для установки достаточно поставить на виртуальную машину или физический сервер OpenScaler версии 23.03. Модуль ядра nfs_multipath будет доступен уже «из коробки»

Нам будет достаточно только убедиться, что он загружен в память:

```
nfs-server# modprobe nfs_multipath
nfs-server# lsmod | grep nfs
nfs_multipath        36864  0
nfs                 544768  1 nfs_multipath
fscache             380928  1 nfs
nfsd                741376  5
auth_rpcgss         159744  2 nfsd,rpcsec_gss_krb5
nfs_acl             16384  1 nfsd
lockd               135168  2 nfsd,nfs
grace               16384  2 nfsd,lockd
sunrpc              704512  27
 nfs_multipath,nfsd,rpcrdma,auth_rpcgss,lockd,rpcsec_gss_krb5,nfs_acl,nfs
```

Схема стенда тестирования

Наш NFS сервер в проводимом тестировании имеет следующие ip адреса на сетевых интерфейсах:

```
enp7s0: 172.17.40.201/24
enp8s0: 172.17.50.201/24
enp9s0: 172.17.60.201/24
```

На NFS клиенте мы настроим следующий адрес:

```
enp7s0: 172.17.4.210/24
```

Для удобства понимания приведём на рисунке 24 схему сети нашего теста тестирования.

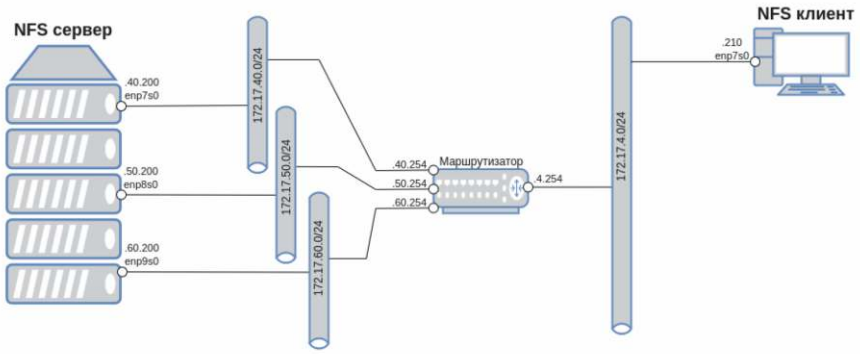

Рисунок 24. Схема стендирования технологии NFS Multipathing

Настраиваем маршрутизацию на стенде тестирования так, чтобы наш NFS сервер был доступен клиенту по всем трём ip адресам. Для этого можно воспользоваться утилитой iproute.

Прописываем маршруты на NFS сервере:

```
nfs-server# ip rule add from 172.17.40.200 lookup 200
nfs-server# ip route add 172.17.4.0/24 via 172.17.40.254 dev enp7s0 table 200

nfs-server# ip rule add from 172.17.50.200 lookup 300
nfs-server# ip route add 172.17.4.0/24 via 172.17.50.254 dev enp8s0 table 300

nfs-server# ip rule add from 172.17.60.200 lookup 400
nfs-server# ip route add 172.17.4.0/24 via 172.17.60.254 dev enp9s0 table 400
```

Прописываем маршруты на NFS клиенте:

```
nfs-client# ip r add 172.17.40.0/24 via 172.17.4.254
nfs-client# ip r add 172.17.50.0/24 via 172.17.4.254
nfs-client# ip r add 172.17.60.0/24 via 172.17.4.254
```

Для экспортирования ресурса **/data** на NFS сервере мы редактируем файл **/etc/exports** и вносим в него строки:

```
/data *(rw,no_root_squash)
```

Тем самым не ограничивая никого в монтировании ресурса с уровнем доступа чтение-запись.

Для удобства тестирования (чтобы не перегружать статью побочной информацией) мы отключили firewall и на сервере и на клиенте.

Запускаем NFS сервер:

```
nfs-server# systemctl enable --now nfs-server.service
```

На NFS клиенте монтируем каталог:

```
nfs-client# mount -t nfs -o
localaddrs=172.17.4.210,remoteaddrs=172.17.40.200~172.17.50.200~172.17.60.200
172.17.40.200:/data /mnt/
```

Из этой команды видно, что перечисление ip адресов сервера и клиента идёт при помощи разделителя «~».

Посмотрим статус смонтированного каталога, для этого на клиенте даём команду:

```
nfs-client# cat /proc/fs/nfs/mutipath/conn_info
==============================Id:2 count 3 proto 4 start=======================
index:0, client_ip:172.17.4.210, server_ip:172.17.40.200, status:connect, load:16
index:1, client_ip:172.17.4.210, server_ip:172.17.50.200, status:connect, load:0
index:2, client_ip:172.17.4.210, server_ip:172.17.60.200, status:connect, load:0
======================================Id:2
end================================
```

Из этого вывода мы видим, что клиент пытается смонтировать NFS ресурс по схеме один-ко-многим, то есть с каждого своего адреса он пытается подключиться ко всем указанным серверным адресам. В данный момент у данной реализации NFS multipathing ограничение на 8 подключений к NFS ресурсу.

Проверка работы

Попробуем отключить на NFS-сервере первый интерфейс:

```
nfs-server# ip l set dev enp7s0 down
```

Проверяем на клиенте:

```
nfs-client # cat /proc/fs/nfs/mutipath/conn_info
==============================Id:1 count 3 proto 4 start=======================
index:0, client_ip:172.17.4.210, server_ip:172.17.40.200, status:disconnect, load:19
index:1, client_ip:172.17.4.210, server_ip:172.17.50.200, status:connect, load:6
index:2, client_ip:172.17.4.210, server_ip:172.17.60.200, status:connect, load:0
======================================Id:1
end================================
```

Файлы в каталоге **/mnt** доступны, мы спокойно можем создавать, удалять, изменять файлы в этом каталоге.

Попробуем отключить ещё один интерфейс:

```
nfs-server# ip l set dev enp8s0 down
```

и проверяем на NFS-клиенте:

```
nfs-client# cat /proc/fs/nfs/mutipath/conn_info
================================Id:1 count 3 proto 4 start=======================
index:0, client_ip:172.17.4.210, server_ip:172.17.40.200, status:disconnect, load:19
index:1, client_ip:172.17.4.210, server_ip:172.17.50.200, status:disconnect, load:7
index:2, client_ip:172.17.4.210, server_ip:172.17.60.200, status:connect, load:5
=======================================Id:1
 end================================
```

Видно, что клиент фиксирует, что для доступа к NFS серверу у него остался только один сетевой линк, но при всём при этом сетевой ресурс NFS доступен и продолжает работать.

Возвращаем обратно в строй наши выключенные на NFS сервере линки и снова проверяем на NFS клиенте:

```
nfs-client# cat /proc/fs/nfs/mutipath/conn_info
================================Id:1 count 3 proto 4 start=======================
index:0, client_ip:172.17.4.210, server_ip:172.17.40.200, status:connect, load:57
index:1, client_ip:172.17.4.210, server_ip:172.17.50.200, status:connect, load:8
index:2, client_ip:172.17.4.210, server_ip:172.17.60.200, status:connect, load:43
=======================================Id:1
 end================================
```

Выводы

Сегодня мы рассмотрели достаточно интересную, новую технологию, которая устраняет некоторые проблемы, которые могут возникать при создании высокодоступных NFS серверов и надеемся, что в будущем, получив развитие, эта технология по праву сможет занять своё место среди решений предоставляющих возможность гибкой и удобной работы с использованием старого и доброго протокола NFS.

9.1.3 SysCare

Устранение уязвимостей в ПО происходит путём обновления версии установленного ПО с последующим перезапуском процессов. В версии openScaler 22.03 SP1 была добавлена технология SysCare, которая поможет произвести установку патча для процесса пользователя без его перезапуска. На рисунке 25 представлен функционал решения.

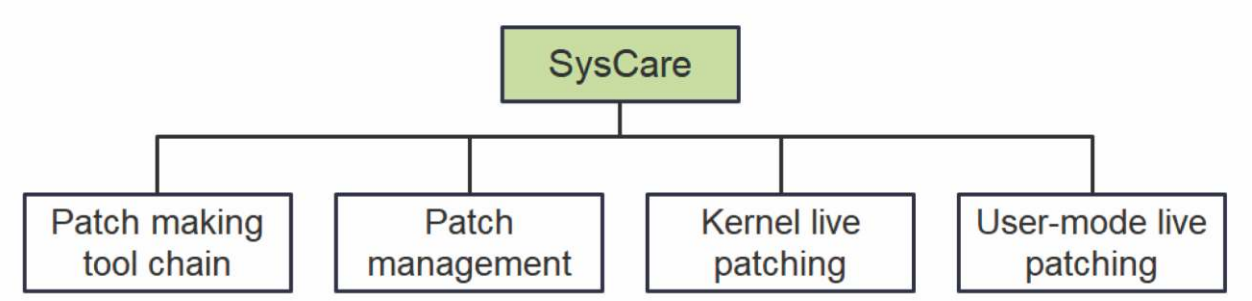

Рисунок 25. Функционал решения SysCare

В официальной документации на сайте https://docs.openeuler.org/en/docs/22.03_LTS_SP1/docs/SysCare/SysCare_introduction.html приведены следующие возможности утилиты:

- Простое создание патчей, как для ядра, так и для уровня пользователя.
- Интерфейс для управления патчами, включая их установку, активацию, деактивация и удаление.

Используются следующие возможности:

- Унифицированные патчи: SysCare скрывает разницу в деталях создания патча, предоставляя унифицированный инструмент для улучшения эффективности эксплуатации и обслуживания.
- Применение патчей на лету для приложения уровня пользователя: SysCare поддерживает наложение патчей на многопоточные и много-процессные приложения в пространстве пользователя, что позволяет достичь эффекта без перезапуска процесса или потока.
- Ленивый механизм: SysCare отлавливает все системные вызовы через ptrace и ждёт их завершения, что позволяет повысить эффективность успешного применения.

Для проверки возможности SysCare будет произведено создание и применение патча для redis, процесса, который хранит данные в памяти и который не желательно перезапускать.

В качестве примера возьмём CVE-2022-24834 https://github.com/redis/redis/security/advisories/GHSA-p8x2-9v9q-c838, которое было исправлено в версии 6.2.13 и попробуем создать патч для версии 6.2.7, которая имеется в составе openScaler 22.03 SP2.

Первым делом установим openScaler и redis на ВМ или сервер. Процесс установки ОС опустим, ввиду его тривиальности.

Для установки redis версии 6.2.7 необходимо установить пакет redis6:

dnf install redis6

На рисунке 26 представлен процесс установки пакета redis6.

```
[root@syscare2 ~]# dnf install redis6
Last metadata expiration check: 0:36:38 ago on Пт 21 июл 2023 09:20:01.
Dependencies resolved.
================================================================================
 Package              Architecture          Version                  Repository                  Size
================================================================================
Installing:
 redis6               x86_64                6.2.7-2.os2203sp2        everything                 1.1 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 1.1 M
Installed size: 4.2 M
Is this ok [y/N]: y
Downloading Packages:
redis6-6.2.7-2.os2203sp2.x86_64.rpm                          3.0 MB/s | 1.1 MB     00:00
--------------------------------------------------------------------------------
Total                                                        3.0 MB/s | 1.1 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                             1/1
  Running scriptlet: redis6-6.2.7-2.os2203sp2.x86_64                             1/1
  Installing       : redis6-6.2.7-2.os2203sp2.x86_64 [================================================================] 1/1[
 2118.112209][ T4925] capability: warning: `dnf' uses 32-bit capabilities (legacy support in use)
  Installing       : redis6-6.2.7-2.os2203sp2.x86_64                             1/1
  Running scriptlet: redis6-6.2.7-2.os2203sp2.x86_64                             1/1
[ 2118.230754][ T4962] systemd-rc-local-generator[4962]: /etc/rc.d/rc.local is not marked executable, skipping.
  Verifying        : redis6-6.2.7-2.os2203sp2.x86_64                             1/1

Installed:
  redis6-6.2.7-2.os2203sp2.x86_64

Complete!
```

Рисунок 26. Процесс установки пакета redis6

Запустим сервис и проверим, что он работает:

systemctl start redis
journalctl -f

```
[root@syscare2 ~]# systemctl start redis
[root@syscare2 ~]# journalctl -f
июл 21 09:56:47 syscare2 dnf[4969]: OS                                         28 kB/s | 3.8 kB     00:00
июл 21 09:56:47 syscare2 dnf[4969]: everything                                 18 kB/s | 3.8 kB     00:00
июл 21 09:56:47 syscare2 dnf[4969]: update                                     27 kB/s | 3.0 kB     00:00
июл 21 09:56:47 syscare2 dnf[4969]: Metadata cache created.
июл 21 09:56:47 syscare2 systemd[1]: dnf-makecache.service: Deactivated successfully.
июл 21 09:56:47 syscare2 systemd[1]: Finished dnf makecache.
июл 21 09:57:01 syscare2 systemd[1]: systemd-hostnamed.service: Deactivated successfully.
июл 21 09:58:22 syscare2 systemctl[4979]: [systemctl start redis] called by PID 4856 (-bash)
июл 21 09:58:22 syscare2 systemd[1]: Starting Redis persistent key-value database...
июл 21 09:58:22 syscare2 systemd[1]: Started Redis persistent key-value database.
```

Рисунок 27. Журнал работы сервиса

Теперь настало время SysCare. В настоящее время, ещё нет готового rpm пакета с SysCare, писать свой мы не будем. Устанавливать будем путём сборки из исходных текстов. Установим необходимые инструменты для сборки:

dnf install -y elfutils-libelf-devel openssl-devel dwarves flex python3-devel rpm-build bison cmake make gcc g++ rust cargo git kernel-devel

```
[root@syscare2 ~]# dnf install elfutils-libelf-devel openssl-devel dwarves flex python3-devel rpm-build bison cm
ake make gcc g++ rust cargo git kernel-devel
Last metadata expiration check: 0:08:36 ago on Пт 21 июл 2023 09:56:47.
Dependencies resolved.
================================================================================================================
 Package                    Architecture   Version                                Repository          Size
================================================================================================================
Installing:
 bison                      x86_64         3.8.2-2.os2203sp2                      OS                 397 k
 cargo                      x86_64         1.64.0-1.os2203sp2                     everything         4.2 M
 cmake                      x86_64         3.22.0-5.os2203sp2                     OS                  11 M
 dwarves                    x86_64         1.22-2.os2203sp2                       everything         119 k
 elfutils-devel             x86_64         0.185-17.os2203sp2                     OS                 373 k
 flex                       x86_64         2.6.4-5.os2203sp2                      OS                 311 k
 gcc                        x86_64         10.3.1-37.os2203sp2                    OS                  29 M
 gcc-c++                    x86_64         10.3.1-37.os2203sp2                    OS                  11 M
 git                        x86_64         2.33.0-12.os2203sp2                    OS                 116 k
 kernel-devel               x86_64         5.10.0-153.12.0.92.os2203sp2           OS                  14 M
 make                       x86_64         1:4.3-4.os2203sp2                      OS                 325 k
 openssl-devel              x86_64         1:1.1.1m-20.os2203sp2                  OS                 1.8 M
 python3-devel              x86_64         3.9.9-24.os2203sp2                     OS                  12 M
 rpm-build                  x86_64         4.17.0-26.os2203sp2                    OS                  71 k
 rust                       x86_64         1.64.0-1.os2203sp2                     everything          24 M
Installing dependencies:
 babeltrace                 x86_64         1.5.8-3.os2203sp2                      OS                 205 k
```

Рисунок 28. Установка дополнительных зависимостей

После установки необходимых пакетов, возьмём исходный код SysCare из git репозитория:

git clone https://gitee.com/openeuler/syscare.git

Так как мы собираем SysCare из исходников, то у нас в системе не будет rpm пакета syscare, значит нам необходимо удалить его требование для установки на этапе создании патчей. Для этого необходимо удалить строку 28 в файле **builder/src/package/rpm_spec_generator.rs:**

Requires: %{syscare_pkg_name}

Пример команд для удаления:

```
cd
syscare
vi builder/src/package/rpm_spec_generator.rs
28gg
dd
:wq
```

```
License:  %{pkg_license}
Summary:  %{pkg_summary}
Requires: %{pkg_requires}
Requires: %{syscare_pkg_name}
"#;

const RPM_SPEC_BODY: &str = r#"
%description
%{pkg_description}
```

Рисунок 29. Исходное состояние файла

```
License:  %{pkg_license}
Summary:  %{pkg_summary}
Requires: %{pkg_requires}
"#;

const RPM_SPEC_BODY: &str = r#"
%description
%{pkg_description}
```

Рисунок 30. Модифицированная строка 28

Создадим каталог сборки и выполним её:

cd syscare
mkdir build
cd build
cmake
-DCMAKE_INSTALL_PREFIX:PATH=/usr ..
make

После успешной сборки необходимо установить собранные утилиты командой:

make install

В процессе установки также устанавливается модуль ядра **upatch.ko**, но не по правильному пути, для возможности его загрузки, он должен располагаться в каталоге **/lib/modules/5.10.0-153.12.0.92.os2203sp2.x86_64/**, поместить его туда можно простым копированием:

cp /usr/libexec/syscare/upatch.ko /lib/modules/5.10.0-153.12.0.92.os2203sp2.x86_64/

Для обновления дерева моделей даём команду

depmod
modinfo upatch

```
[root@syscare2 build]# cp /usr/libexec/syscare/upatch.ko /lib/modules/5.10.0-153.12.0.92.os2203sp2.x86_64/
[root@syscare2 build]# depmod
[root@syscare2 build]# modinfo upatch
filename:       /lib/modules/5.10.0-153.12.0.92.os2203sp2.x86_64/upatch.ko
version:        dev-9e34248
license:        GPL
description:    kernel module for upatch(live-patch in userspace)
author:         Zongwu Li (lzw32321226@163.com)
author:         Longjun Luo (luolongjuna@gmail.com)
srcversion:     B746CAEEF2AB7A79EF19E2E
depends:
retpoline:      Y
name:           upatch
vermagic:       5.10.0-153.12.0.92.os2203sp2.x86_64 SMP mod_unload modversions
```

Рисунок 31. Параметры модуля ядра upatch.ko

Как видно, ядро успешно увидело модуль, теперь его можно загрузить:

modprobe upatch
lsmod | grep upatch
dmesg| grep upatch

```
[root@syscare2 build]# modprobe upatch
[root@syscare2 build]# lsmod | grep upatch
upatch                 73728  0
[root@syscare2 build]# dmesg | grep upatch
[ 4559.526663] upatch: loading out-of-tree module taints kernel.
[ 4559.527610] upatch: module verification failed: signature and/or required key missing - tainting kernel
[ 4559.535137] upatch - dev-9e34248 load successfully
```

Рисунок 32. Успешно загруженный модуль upatch

Модуль успешно загрузился. Теперь необходимо подготовить патч и собрать его.

Патч пожно взять из коммита в репозитории redis. Для этого открываем репозиторий redis на github и переключаемся на ветку версии 6.2 как показано на рисунке 33.

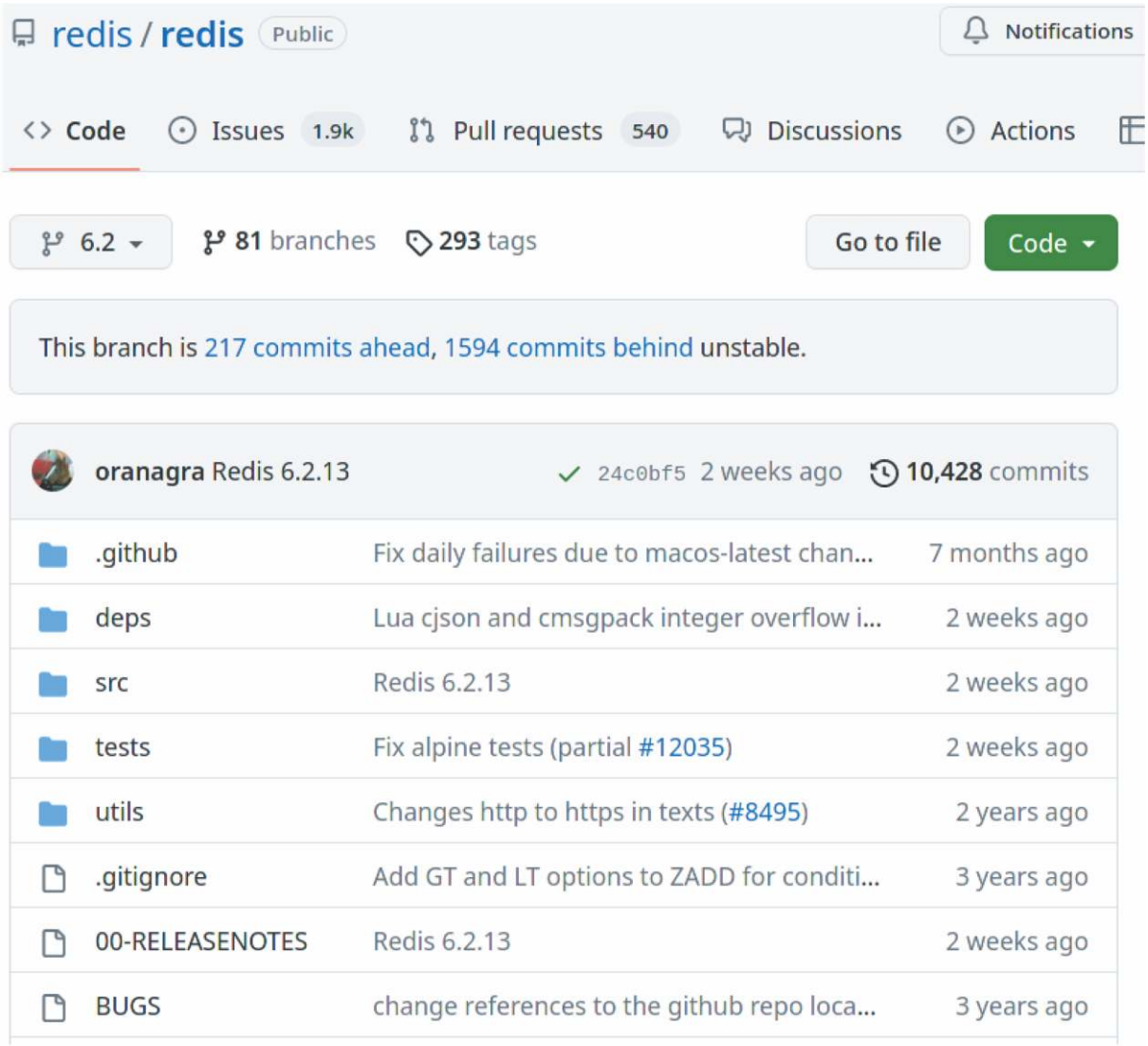

Рисунок 33. redis на guthub, ветка 6.2

Открываем список коммитов и ищем коммит с исправлением как показано на рисунке 34.

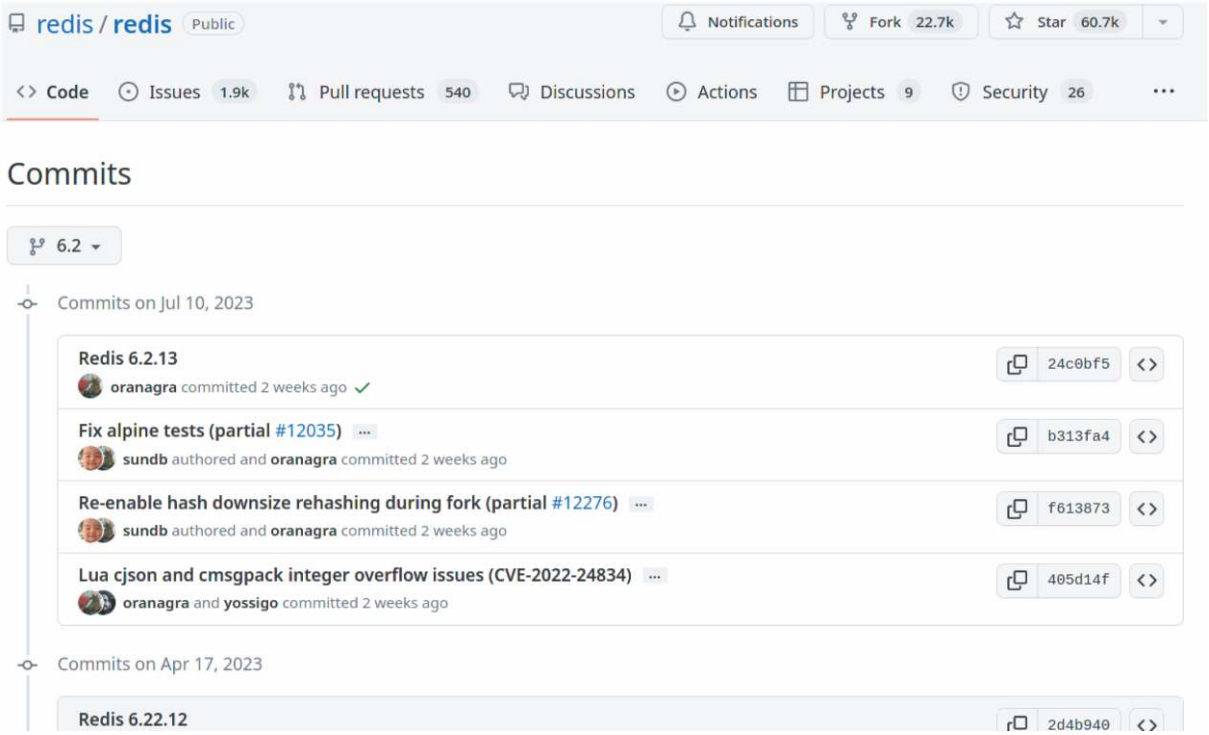

Рисунок 34. Список коммитов в redis ветки 6.2

Искомый коммит имеет в своём комментарии CVE-2022-24834 и имеет хеш 405d14fd44c6a159680484f048b6811e6af21143, открываем его.

Чтобы получить этот коммит в виде patch файла, необходимо в адресной строке добавить .patch, тогда путь будет выглядеть так:
https://github.com/redis/redis/commit/405d14fd44c6a159680484f048b6811e6af21143.patch

Его можно сохранить как через браузер, так и выкачать через curl или wget:
cd ~
mkdir redis-patch
cd redis-patch
curl
https://github.com/redis/redis/commit/405d14fd44c6a159680484f048b6811e6af21143.patch -o CVE-2022-24834.patch

Дальше необходимы пакеты с исходниками и отладочной информацией: **redis6-6.2.7-2.os2203sp2.src.rpm** и **redis6-debuginfo-6.2.7-2.os2203sp2.x86_64.rpm**. Получить их можно через репозитории source и debug. По умолчанию эти репозитории отключены, но их можно включить. Получим список всех доступных репозиториев:

dnf repolist –all

```
[root@syscare2 redis-patch]# dnf repolist --all
repo id                                     repo name                                   status
EPOL                                        EPOL                                        disabled
OS                                          OS                                          enabled
debuginfo                                   debuginfo                                   disabled
everything                                  everything                                  enabled
source                                      source                                      disabled
update                                      update                                      enabled
update-source                               update-source                               disabled
```

Рисунок 35. Список доступных репозиториев

Включим необходимые репозитории:

dnf config-manager --set-enabled source
dnf config-manager --set-enabled debuginfo
dnf repolist --all

Скачаем требуемые пакеты:

dnf download --source redis6
dnf download redis6-debuginfo

Все подготовительные шаги выполнены и можно приступить к сборке патча.
Для сборки необходимо выполнить команду:

syscare build -n redis_CVE-2022-24834 –source redis6-6.2.7-2.os2203sp2.src.rpm –debuginfo redis6-debuginfo-6.2.7-2.os2203sp2.x86_64.rpm CVE-2022-24834.patch

И-и-и, провал…

```
name:        redis_CVE-2022-24834
version:     1
release:     1
arch:        x86_64
type:        UserPatch
target:      redis6-6.2.7-2.os2203sp2
target_elf:  (none)
license:     BSD and MIT
description: (none)
patch:
1. CVE-2022-24834.patch
------------------------------
Checking build parameters
Pareparing to build patch
Building patch, this may take a while
Error: Process "/usr/libexec/syscare/upatch-build" exited unsuccessfully, exit_code=253
For more information, please check "/root/redis-patch/syscare-build.9767/build.log"
```

Рисунок 36. Ошибка компиляции

Как же так? Или что-то забыли. Время читать логи:

less /root/redis-patch/syscare-build.9767/build.log

Как видно, не хватает пакетов необходимых для сборки redis. Попробуем получить и установить требуемые пакеты:

rpmbuild -ra redis6-6.2.7-2.os2203sp2.src.rpm

Видно, что нужны пакеты **libatomic** и **systemd-devel**, установим их командой:

dnf install systemd-devel libatomic

Пробуем ещё раз:

syscare build -n redis_CVE-2022-24834 –source redis6-6.2.7-2.os2203sp2.src.rpm –debuginfo redis6-debuginfo-6.2.7-2.os2203sp2.x86_64.rpm CVE-2022-24834.patch

На выходе получился rpm файл, попробуем поставить:

dnf localinstall patch-redis6-6.2.7-2.os2203sp2-redis_CVE-2022-24834-1-1.x86_64.rpm

И проверим патчи:

syscare list

Теперь осталось применить патч:

syscare apply redis6-6.2.7-2.os2203sp2/redis_CVE-2022-24834
syscare list

Посмотрим, что в системном журнале:

journalctl

Видно что патч применился.

Теперь настала пора проверить: а работает ли он? Для этого воспользуемся exploit: **https://github.com/convisolabs/CVE-2022-24834.**

Проверку работоспособности проведем сразу после перезагрузки системы, так сказать с чистого листа. Производим перезагрузку, но syscare не активируем.

Скачаем expoit и установим зависимости:

git clone https://github.com/convisolabs/CVE-2022-24834
pip3 install pwn
dnf install nc

Выполним exploit:

cd CVE-2022-24834
python3 redis_cve-2022-24834.py

```
[root@syscare2 ~]# cd CVE-2022-24834/
[root@syscare2 CVE-2022-24834]# python3 redis_cve-2022-24834.py
[*] Starting server on 82.204.182.121:4444
[*] Sending evil lua script to localhost:6379
[+] Opening connection to localhost on port 6379: Done
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Traceback (most recent call last):
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 111, in <module>
    main()
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 104, in main
    r = redis.EVAL(evil_lua_script)
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 59, in EVAL
    return self.cmd(["EVAL", script, "0"])
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 33, in cmd
    return self.recv()
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 37, in recv
    t = self.conn.recv(1)
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 105, in recv
    return self._recv(numb, timeout) or b''
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 175, in _recv
    if not self.buffer and not self._fillbuffer(timeout):
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 154, in _fillbuffer
    data = self.recv_raw(self.buffer.get_fill_size())
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/sock.py", line 56, in recv_raw
    raise EOFError
EOFError
^CException ignored in: <module 'threading' from '/usr/lib64/python3.9/threading.py'>
Traceback (most recent call last):
  File "/usr/lib64/python3.9/threading.py", line 1470, in _shutdown
    lock.acquire()
KeyboardInterrupt:
[*] Closed connection to localhost port 6379
[root@syscare2 CVE-2022-24834]#
```

Рисунок 37. Выполнение exploit завершено ошибкой

Как видно, выполнение exploit завершилось ошибкой, а как там redis?

systemctl status redis

```
[root@syscare2 CVE-2022-24834]# systemctl status redis
× redis.service - Redis persistent key-value database
     Loaded: loaded (/usr/lib/systemd/system/redis.service; disabled; vendor preset: disabled)
    Drop-In: /etc/systemd/system/redis.service.d
             └─limit.conf
     Active: failed (Result: core-dump) since Fri 2023-08-18 10:16:09 MSK; 1min 4s ago
    Process: 4973 ExecStart=/usr/bin/redis-server /etc/redis/redis.conf --daemonize no --supervised systemd (code=dumped, signal=SEGV)
   Main PID: 4973 (code=dumped, signal=SEGV)
      Status: "Ready to accept connections"

авг 18 10:13:12 syscare2 systemd[1]: Starting Redis persistent key-value database...
авг 18 10:13:12 syscare2 systemd[1]: Started Redis persistent key-value database.
авг 18 10:16:09 syscare2 systemd[1]: redis.service: Main process exited, code=dumped, status=11/SEGV
авг 18 10:16:09 syscare2 systemd[1]: redis.service: Failed with result 'core-dump'.
[root@syscare2 CVE-2022-24834]#
```

Рисунок 38. Журнал работы redis

Упал, посмотрим в логах, узнаем что произошло:

tail -n 20 /var/log/redis/redis.log

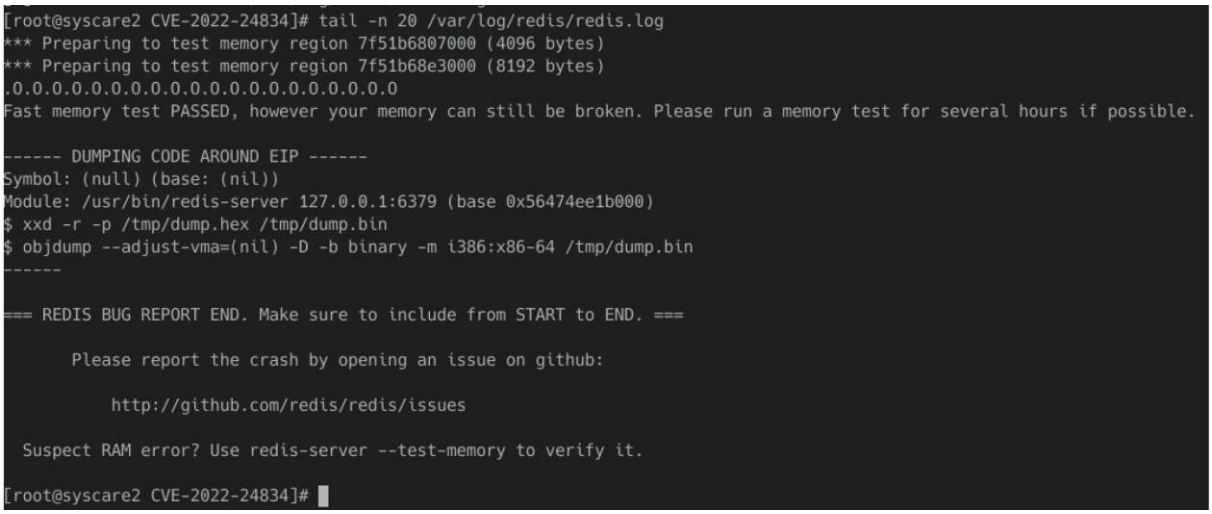

Рисунок 39. Ошибка возникшая в ходе работы redis

Видно что произошло повреждение памяти, посмотрим детальнее, чем это вызвано:

less /var/log/redis/redis.log

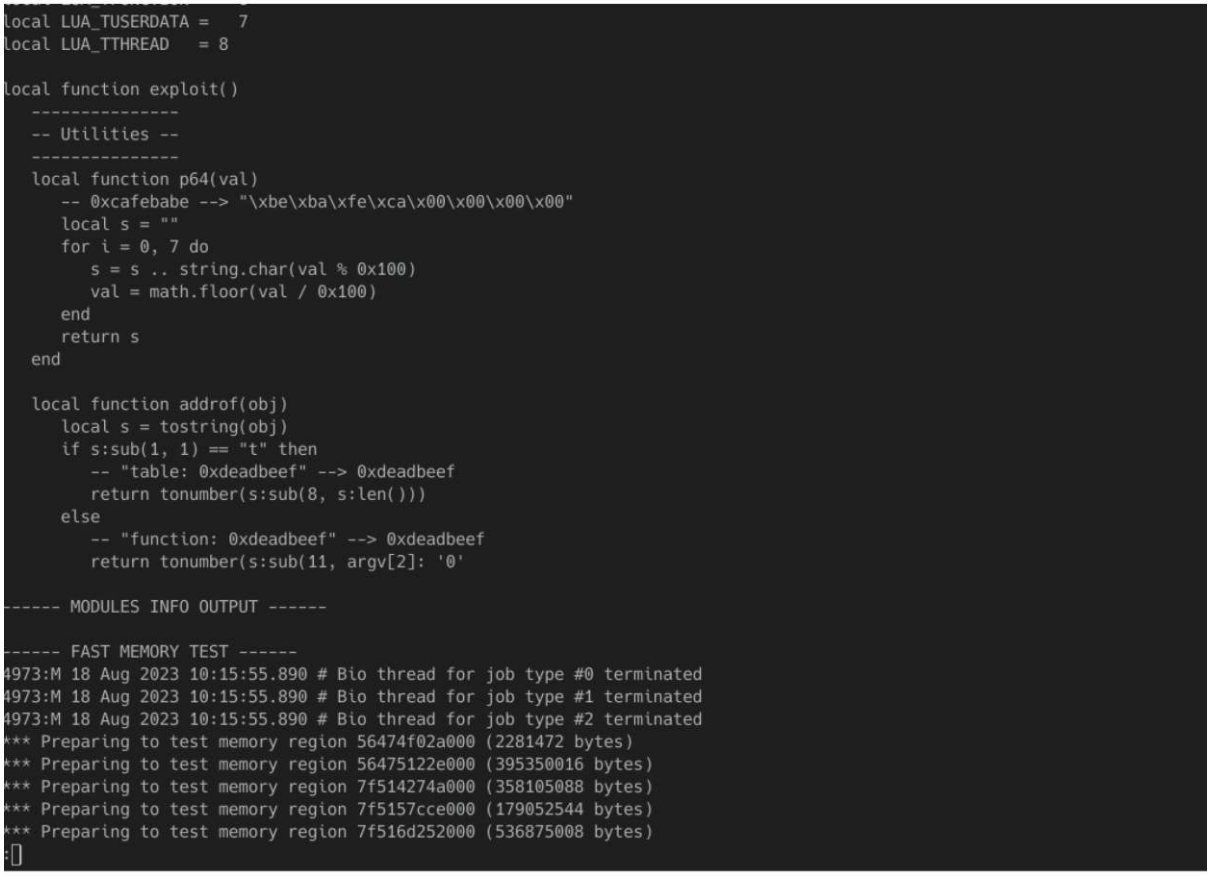

Рисунок 40. Результат работы эксплоита

Ага, это сработал exploit. Значит данный exploit вызывает падение redis и может привести к DoS. Посмотрим как он поведёт себя при применённом патче. Загрузим модуль ядра и проверим состояние syscare:

modprobe upatch

syscare list

```
[root@syscare2 CVE-2022-24834]# modprobe upatch
[root@syscare2 CVE-2022-24834]# syscare list
Uuid                                 Name                                                Status
41f4efba-0c92-41fc-b14a-75a469b2effb     redis6-6.2.7-2.os2203sp2/redis_CVE-2022-24834 NOT-APPLIED
[root@syscare2 CVE-2022-24834]#
```

Рисунок 41. Загрузка модулей upatch и list

Перезапускаем процесс redis и применяем патч:

systemctl restart redis

syscare apply redis6-6.2.7-2.os2203sp2/redis_CVE-2022-24834

syscare list

```
[root@syscare2 CVE-2022-24834]# systemctl restart redis
[root@syscare2 CVE-2022-24834]# syscare apply redis6-6.2.7-2.os2203sp2/redis_CVE-2022-24834
[root@syscare2 CVE-2022-24834]# syscare list
Uuid                                 Name                                                Status
41f4efba-0c92-41fc-b14a-75a469b2effb     redis6-6.2.7-2.os2203sp2/redis_CVE-2022-24834 ACTIVED
[root@syscare2 CVE-2022-24834]#
```

Рисунок 42. Перезапуск сервиса redis и применение патча через SysCare

Приготовления выполнены, теперь в бой:

python3 redis_cve-2022-24834.py

```
[root@syscare2 CVE-2022-24834]# python3 redis_cve-2022-24834.py
[*] Starting server on 82.204.182.121:4444
[*] Sending evil lua script to localhost:6379
[←] Opening connection to localhost on port 6379
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
[+] Opening connection to localhost on port 6379: Done
Traceback (most recent call last):
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 111, in <module>
    main()
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 104, in main
    r = redis.EVAL(evil_lua_script)
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 59, in EVAL
    return self.cmd(["EVAL", script, "0"])
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 33, in cmd
    return self.recv()
  File "/root/CVE-2022-24834/redis_cve-2022-24834.py", line 37, in recv
    t = self.conn.recv(1)
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 105, in recv
    return self._recv(numb, timeout) or b''
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 175, in _recv
    if not self.buffer and not self._fillbuffer(timeout):
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/tube.py", line 154, in _fillbuffer
    data = self.recv_raw(self.buffer.get_fill_size())
  File "/usr/local/lib/python3.9/site-packages/pwnlib/tubes/sock.py", line 56, in recv_raw
    raise EOFError
EOFError
^CException ignored in: <module 'threading' from '/usr/lib64/python3.9/threading.py'>
Traceback (most recent call last):
  File "/usr/lib64/python3.9/threading.py", line 1470, in _shutdown
    lock.acquire()
KeyboardInterrupt:
[*] Closed connection to localhost port 6379
[root@syscare2 CVE-2022-24834]#
```

Рисунок 43. Результат выполнения эксплоита

Exploit опять сработал… Что-то не так… Посмотрим системный журнал:

journalctl -f

Ага, дело в selinux. Есть два пути: отключить selinux и правильный. Пойдём по правильному пути. Установим пакет setroubleshoot:

dnf install setroubleshoot

Найдём в **audit.log** событие, на базе которого создадим модуль selinux:

grep denied /var/log/audit/audit.log

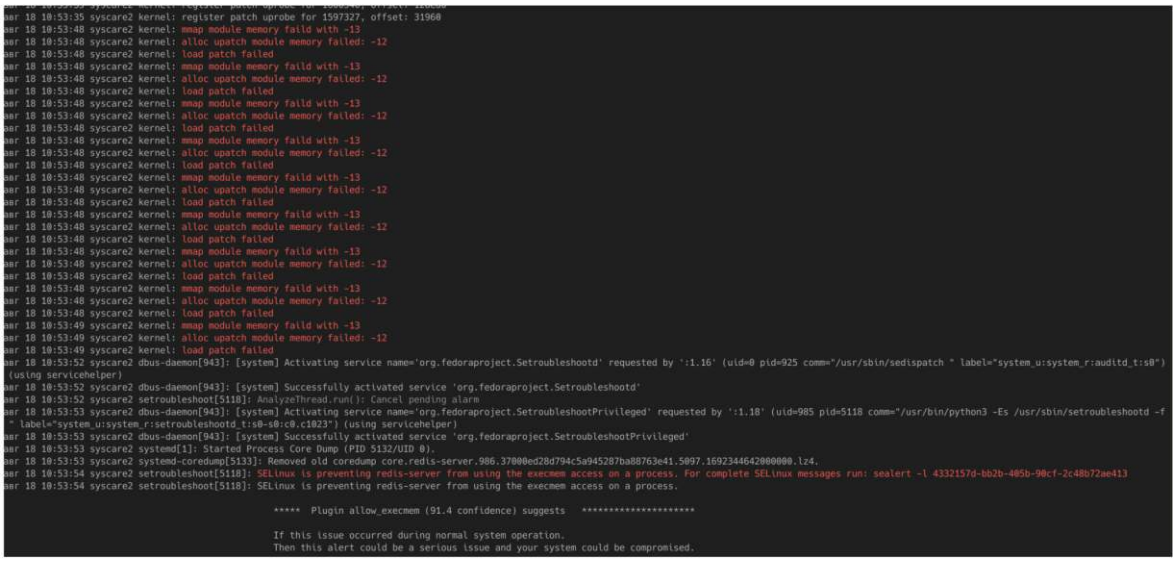

Рисунок 44. Блокировка от Selinux

Возьмём последнее событие:

grep 1692345229.419:115 /var/log/audit/audit.log | audit2allow -a -M syscare-redis

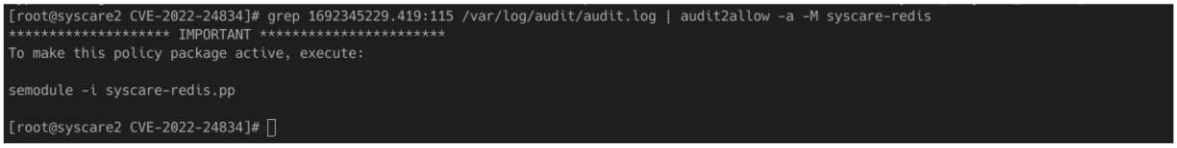

Рисунок 45. Исполнение audit2allow

Загрузим созданный модуль:

semodule -i syscare-redis.pp

Перезагрузим redis:

systemctl restart redis

И попробуем ещё раз:

python3 redis_cve-2022-24834.py

Прошло довольно много времени, а exploit и не думает срабатывать, как представлено на рисунке 46.

Рисунок 46. Сервис redis продолжает работу после применения эксплоита

Таким образом, созданный патч для syscare и вправду работает. Необходимо только не забыть про SElinux.

В заключение можно сказать, что данный инструментарий является одним из ключевых новых веяний в инструментарии управления информационной безопасности и защиты корпоративной ИТ инфраструктуры. И заслуживает пристального внимания и анализа со стороны инженеров подразделений обеспечения ИБ и служб технической поддержки и сопровождения корпоративных сервисов.

9.1.4 A-Tune

С зарождения человечества перед людьми в том или ином виде стояла проблема хранения и обработки информации – начиная от Тэртерийских глиняных табличек, датируемых V тыс. до н.э, великой библиотеки в Александрии и заканчивая сегодняшними возможностями ИТ-решений и технологий, от простых бытовых записей (например, учета съестных ресурсов) в древности до сложной бизнес-аналитики и систем принятия решений на производственных предприятиях сегодня. Каждая эпоха привносила свои знания и стимулировала все больший рост объемов информации.

Очередной “эпохой” осуществившей очередной количественный скачок по объему хранимой информации и формированию новых требований к оперативности ее обработки можно считать начавшуюся, казалось бы, не так давно эру сети Интернет и последовавшие за последние десятилетия скачки темпов роста хранимых данных (во многом благодаря увеличению объемов веб-контента).

Рисунок 47. Рост объемов хранимых данных

Разнотипность приложений, сервисов и типов обрабатываемых ими данных приводит к необходимости использования диверсифицированных вычислений. Иначе количество разнотипных специализированных операционных систем и программного обеспечения будет таковым, что процесс управления и администрирования типового ИТ ландшафта будет крайне переусложнен и приведет к росту числа ИТ персонала ответственного за эксплуатацию серверных систем компании, необходимости знания дополнительных и/или специализированных решений а также сильно повысит риск человеческой ошибки при настройке систем.

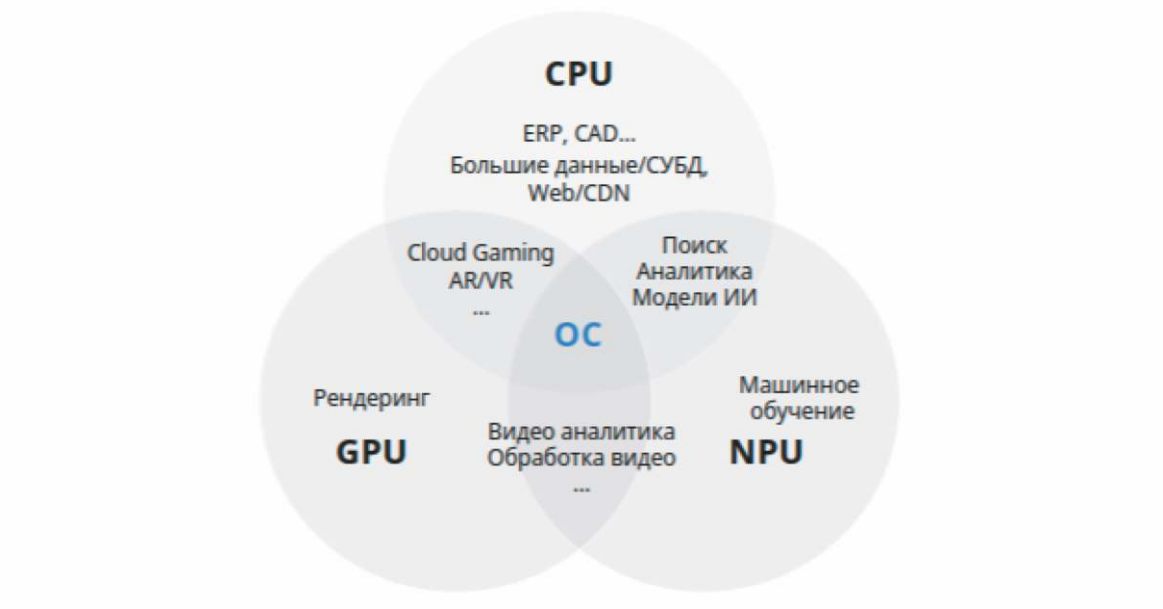

Рисунок 48. Единая ОС для всех сценариев использования - основа диверсифицированных вычислений

Но если вопрос унификации операционной системы в рамках диверсифицированных вычислений и можно решить, то проблема необходимости настройки большого количества типовых серверов, ориентированных на различные типы загрузки в зависимости от предоставляемых ими сервисов и решаемых в ИТ инфраструктуре заказчика задач к сожалению остается. Зачастую именно настройка типовых систем оказывается основным узким местом влияющим на производительность всей экосистемы в целом. Действительно ли все сервера ответственные за исполнение определенных задач, к примеру, сервера СУБД и типы их загрузки настолько идентичны что для них подойдет единый профиль настройки? Есть ли возможность поручить штатному администратору СУБД анализ каждого типового сервера коих могут быть десятки или даже сотни?

Искоренить подобные вопросы и максимально упростить задачу оптимизации настроек каждого сервера в отдельности в зависимости от типа его загрузки и реалий эксплуатации призван инструмент A-Tune интегрированный в наш дистрибутив OpenScaler, являющийся локализованной версией китайского флагмана свободного программного обеспечения под названием openEuler.

A-Tune – Представляет собой автоматизированную систему анализа типа нагрузки и оптимизации производительности сервера в зависимости от профилей его нагрузки, используемого программного обеспечения и условий его эксплуатации. Главным “секретным оружием” данного продукта является наличие инструментария позволяющего оптимизировать производительность каждой конкретной серверной системы на основе искусственного интеллекта. С помощью технологий искусственного интеллекта A-Tune имеет возможность определить наиболее подходящий “профиль” использования системы, провести проверку текущих параметров конфигурационных файлов приложения и настроек ОС на предмет их оптимальности при данных условиях эксплуатации и в конечном счете предоставить системному администратору вывод-отчет содержащий рекомендации по изменению настроек системы. На основе данного отчета системным администратором могут быть внесены изменения в настройки с целью достижения оптимальной производительности и надежности работы системы. Впоследствии, сформированный итоговый “профиль” системы может быть растиражирован на другие сервера.

В реальной IT среде любого коммерческого или государственного предприятия могут применяться сотни и даже тысячи профилей (сценариев эксплуатации систем), и каждый из них требует большого числа программных и аппаратных конфигураций для ресурсов вычисления и хранения, сетевых ресурсов. В лабораторных условиях невозможно предусмотреть все приложения, бизнес-сценарии и комбинации аппаратных средств. Именно для решения данной проблематики сообщество openEuler разработало приложение A-Tune.

Рисунок 49. Ключевые сферы применения системы A-Tune

На рисунке ниже показана основная техническая архитектура A-Tune, состоящая из модуля интеллектуального принятия решений, системного профиля и системы взаимодействия.

Уровень интеллектуального принятия решений состоит из подсистемы осведомленности, которая выполняет функции интеллектуального анализа и оповещения о состоянии приложений, и подсистемы принятия решений, которая отвечает за фактическую оптимизацию и внесение изменений в конфигурацию системы.

Уровень системного профиля состоит из модуля компоновки функций и двухуровневой модели классификации. Модуль компоновки функций служит для автоматического выбора сервисных функций, а двухуровневая модель классификации используется для обучения и классификации сервисных моделей.

Уровень системы взаимодействия отслеживает и конфигурирует различные системные ресурсы и выполняет политики оптимизации.

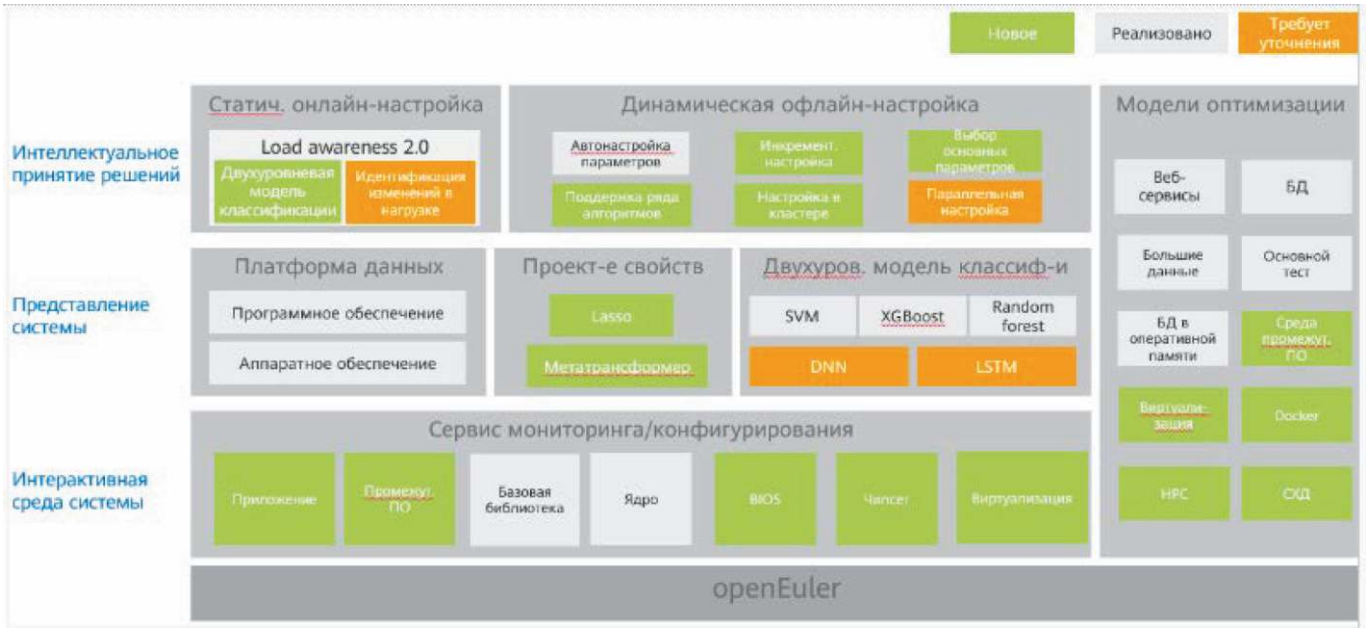

Рисунок 50. Архитектура инструмента A-Tune

Сообществом openEuler и рядом коммерческих компаний была подтверждена эффективность работы инструмента. В таблице 27 приведены основные функции, поддерживаемые приложением A-Tune, их уровень зрелости и рекомендации по использованию.

Таблица 27. Перечень функций и уровень их готовности к использованию в продуктивных системах.

Функция	Уровень готовности	Рекомендации к применению
Автоматическая оптимизация11 наименований программных решений, представляющих 7 различных типов нагрузки	Протестировано	Рекомендовано к применению
Создание пользовательских профилей нагрузки и сервисных моделей	Протестировано	Рекомендовано к применению
Автоматическая оптимизация параметров	Протестировано	Рекомендовано к применению

Исходя из характеристик рабочей нагрузки приложений, A-Tune выделяет 11 типов сервисов. Подробная информация о факторах, снижающих производительность каждого типа сервиса, и о приложениях, поддерживаемых A-Tune, приведена в таблице 28.

Таблица 28. Поддерживаемые типы рабочих нагрузок и приложений для оптимизации

Категория сервиса	Тип	Проблематика	Поддерживаемые приложения
Default	Типо по умолчанию	Низкая загрузка ресурсов процессора, пропускной способности памяти, сети и устройств ввода-вывода.	---
WebServer	Приложение HTTPS	Высокая загрузка ресурсов процессора.	Nginx
Big_Data	Большие Данные	Высокая загрузка ресурсов процессора и устройств ввода-вывода.	Hadoop, Spark

Big_Database	База Данных	Реляционная база данных Чтение: высокая загрузка ресурсов процессора, пропускной способности памяти и сети. Запись: высокая загрузка ресурсов устройств ввода-вывода. Нереляционная база данных Высокая загрузка ресурсов процессора и устройств ввода-вывода.	MongoDB, MySQL, PostgreSQL, MariaDB
In-Memory_ Computing	Приложение с интенсивным потреблением памяти	Высокая загрузка ресурсов процессора и пропускной способности памяти.	SPECjbb2015
In-Memory_ DataBase	Приложение с интенсивным потреблением вычислительных и сетевых ресурсов	Высокая загрузка одноядерного процессора и сетевых ресурсов в многоэкземплярных сценариях.	Redis
Single_Computer_ Intensive_Jobs	Приложение с интенсивным потреблением вычислительных ресурсов	Высокая загрузка ресурсов одноядерного процессора и пропускной способности памяти некоторых подэлементов	SPECCPU2006
Communication	Приложение с интенсивным потреблением сетевых ресурсов	Высокая загрузка ресурсов процессора и сети.	Dubbo
Idle	Система в состоянии ожидания	Система находится в состоянии ожидания, и на ней не запущены никакие приложения.	---

Но теория, это прекрасно, как же это работает на практике? Итак, глазами системного администратора можно увидеть три ключевых режима работы приложения. Давайте рассмотрим их постепенно переходя от простого к сложному:

1. Применение готового “профиля” системы или “статический” режим работы

Тот случай когда системный администратор или IT подразделение ответственное за эксплуатацию системы и сервисов полностью уверены в корректности настроек, “эталонные” параметры могут быть растиражированы и применены на всех типовых системах подходящих под данный профиль.

Для этого системный администратор может воспользовавшись одним из 11 заранее предустановленных “профилей” представленных в таблице 2 написать свой собственный, используя максимально простую и понятную структуру файла описания представленного ниже.

```
профиль web/nginx/https-long-connection.conf (Настройка Nginx)
[main]

include = default-default

[kernel_config]

[bios]

Support SMMU = Disabled

[bootloader.grub2]

[sysfs]

[systemctl]

irqbalance = stop

[sysctl]

net.ipv4.tcp_tw_reuse = 1

net.ipv4.ip_local_port_range = 1024     65500

net.ipv4.tcp_max_tw_buckets = 5000

net.core.somaxconn = 65535

net.core.netdev_max_backlog = 262144

net.ipv4.tcp_max_syn_backlog = 262144

net.ipv4.tcp_fin_timeout = 1
```

```
net.ipv4.tcp_keepalive_time = 60
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
[script]
openssl_hpre = 0
prefetch = off
ethtool =  -X {network} hfunc toeplitz
[ulimit]
{user}.hard.nofile = 102400
{user}.soft.nofile = 102400
[schedule_policy]
[check]
[tip]
SELinux provides extra control and security features to linux kernel. Disabling SELinux will improve the performance but may cause security risks. = kernel
```

Посмотреть готовые профили настройки под конкретные приложения можно при помощи команды:

```
# atune-adm list
Support profiles:
+-----------------------------------------------+-----------+
| ProfileName                                   | Active    |
+===============================================+===========+
| arm-native-android-container-robox            | false     |
+-----------------------------------------------+-----------+
| basic-test-suite-baseline-fio                 | false     |
+-----------------------------------------------+-----------+
| basic-test-suite-baseline-lmbench             | false     |
+-----------------------------------------------+-----------+
```

```
| basic-test-suite-baseline-netperf        | false   |
+-----------------------------------------+-------+
| basic-test-suite-baseline-stream         | false   |
+-----------------------------------------+-------+
| basic-test-suite-baseline-unixbench      | false   |
+-----------------------------------------+-------+
| basic-test-suite-speccpu-speccpu2006     | false   |
+-----------------------------------------+-------+
| basic-test-suite-specjbb-specjbb2015     | false   |
+-----------------------------------------+-------+
| big-data-hadoop-hdfs-dfsio-hdd           | false   |
+-----------------------------------------+-------+
| big-data-hadoop-hdfs-dfsio-ssd           | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-bayesian           | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-kmeans             | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-sql1               | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-sql10              | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-sql2               | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-sql3               | false   |
+-----------------------------------------+-------+
| big-data-hadoop-spark-sql4               | false   |
+-----------------------------------------+-------+
```

```
| big-data-hadoop-spark-sql5              | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-sql6              | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-sql7              | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-sql8              | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-sql9              | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-tersort           | false   |
+-----------------------------------------+---------+
| big-data-hadoop-spark-wordcount         | false   |
+-----------------------------------------+---------+
| cloud-compute-kvm-host                  | false   |
+-----------------------------------------+---------+
| database-mariadb-2p-tpcc-c3             | false   |
+-----------------------------------------+---------+
| database-mariadb-4p-tpcc-c3             | false   |
+-----------------------------------------+---------+
| database-mongodb-2p-sysbench            | false   |
+-----------------------------------------+---------+
| database-mysql-2p-sysbench-hdd          | false   |
+-----------------------------------------+---------+
| database-mysql-2p-sysbench-ssd          | false   |
+-----------------------------------------+---------+
| database-postgresql-2p-sysbench-hdd     | false   |
+-----------------------------------------+---------+
| database-postgresql-2p-sysbench-ssd     | false   |
+-----------------------------------------+---------+
```

```
| default-default                          | true  |
+------------------------------------------+-------+
| docker-mariadb-2p-tpcc-c3                | false |
+------------------------------------------+-------+
| docker-mariadb-4p-tpcc-c3                | false |
+------------------------------------------+-------+
| encryption-AES-chauffeur                 | false |
+------------------------------------------+-------+
| encryption-MD5-chauffeur                 | false |
+------------------------------------------+-------+
| encryption-RSAPublic-chauffeur           | false |
+------------------------------------------+-------+
| hpc-gatk4-human-genome                   | false |
+------------------------------------------+-------+
| in-memory-database-redis-redis-benchmark | false |
+------------------------------------------+-------+
| middleware-dubbo-dubbo-benchmark         | false |
+------------------------------------------+-------+
| storage-ceph-vdbench-hdd                 | false |
+------------------------------------------+-------+
| storage-ceph-vdbench-ssd                 | false |
+------------------------------------------+-------+
| virtualization-consumer-cloud-olc        | false |
+------------------------------------------+-------+
| virtualization-mariadb-2p-tpcc-c3        | false |
+------------------------------------------+-------+
| virtualization-mariadb-4p-tpcc-c3        | false |
+------------------------------------------+-------+
| web-apache-traffic-server-spirent-pingpo | false |
+------------------------------------------+-------+
```

```
| web-nginx-http-long-connection          | false    |
+-----------------------------------+-----+
| web-nginx-http-short-connection         | false    |
+-----------------------------------+-----+
| web-nginx-https-long-connection         | false    |
+-----------------------------------+-----+
| web-nginx-https-short-connection        | false    |
+-----------------------------------+-----+
| web-tomcat-http-connection              | false    |
+-----------------------------------+-----+
```

Если системный администратор видит, что для требуемого приложения уже есть готовый профиль, и предлагаемый им набор настроек устраивает, то он может просто применить его и на этом настройка системы будет окончена.

Применение профиля

К примеру, у нас работает сервер приложений tomcat и мы хотим использовать профиль для настройки параметров ОС применительно к этому приложению. Для этого мы даём команду:

```
# atune-adm profile web-tomcat-http-connection
```

Выбранный профиль будет применен на данном сервере со всем входящим в него набором настроек. Теперь, если мы дадим команду

```
# atune-adm list | grep web-tomcat-http-connection
|        web-tomcat-http-connection              | true     |
```

мы убедимся, что этот профиль является активным на данном сервере.

Тиражирование готового профиля может быть осуществлено либо посредством командной строки и команды вида:

```
# atune-adm profile web-nginx-https-long-connection
```

Либо посредством централизованного веб-интерфейса управления серверами с установленным на них инструментом A-Tune. Внешний вид которого представлен на рисунке 51. Какой бы вариант не предпочел использовать системный администратор – задача выполняется в течении нескольких минут.

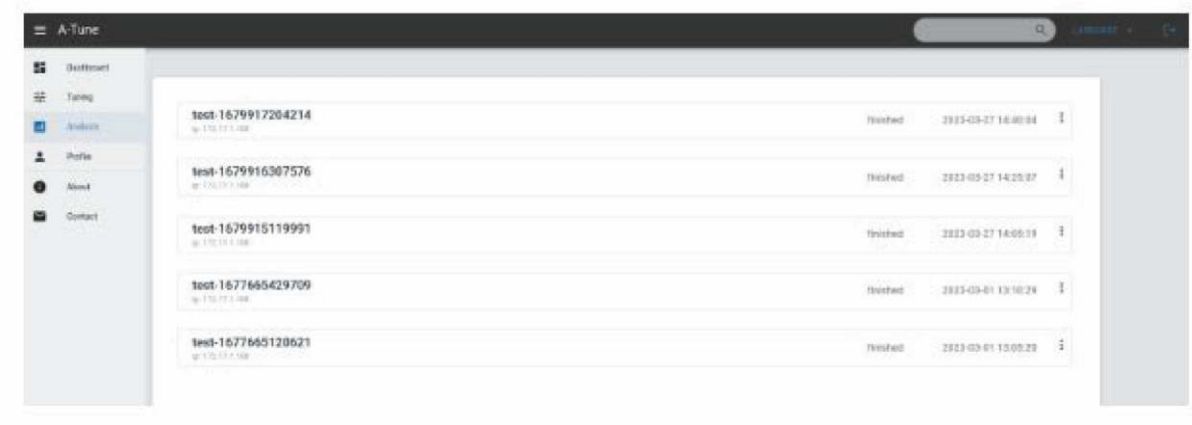

Рисунок 51. внешний вид веб-интерфейса A-Tune

2. Изучение готового “профиля” и автоматизированная корректировка неоптимальных параметров настройки или “динамический” режим работы

А этот случай вполне можно обозвать “доверяй но проверяй”, системным администратором подготовлены как ему кажется оптимальные параметры работы системы и конкретного сервиса на его борту (будь то СУБД, стек решений Hadoop или веб-сервер) но действительно ли они оптимальны? Можно попросить A-Tune это проверить путём перебора значимых параметров в конфигурационном файле этого приложения

A-Tune обеспечивает возможность автоматического поиска оптимальной конфигурации приложения и ОС, избавляя от необходимости ручной настройки его параметров и оценки их производительности. Это значительно повышает эффективность поиска оптимальных конфигураций приложения. Для того, чтобы воспользоваться этой возможностью, надо подготовить так называемый проектный файл (в формате yaml) для требуемого приложения, производительность которого планируется оптимизировать (этот файл впоследствии размещается в каталоге /etc/atuned/tuning/), а также файл конфигурации клиента (также в yaml формате). Файл конфигурации клиента будет применяться для создания повторяемой нагрузки на приложение и количественной оценки работы приложения на каждом прогоне при изменении параметров конфигурации.

В качестве примера, рассмотрим автоматический подбор параметров для такого распространенного сервера СУБД как MySQL

ПРИМЕЧАНИЕ

- Все конфигурационные файлы используемые в рамках данного примера можно получить по адресу https://gitee.com/openeuler/A-Tune/tree/master/examples/tuning/mysql_sysbench
- В качестве тестового сервера используется система с 4 ядрами CPU, 4 Гб оперативной памяти и установленной операционной системой openScaler 22.03 SP1.
- Количество ядер для работы данного примера должно быть не меньше 4, так как при настройке параметров используется команда taskset, которая прикрепляет исполнение процессов к конкретным ядрам CPU и здесь используются все 4 ядра.

Для подготовки тестовой системы и конфигурационных файлов можно воспользоваться готовым скриптом (который поставляется в gitee репозитории, ссылка на который дана в примечании выше) **prepare.sh**

Если рассмотреть этот скрипт, то можно заметить, что в нём происходит установка пакетов mysql сервера (при этом конфигурационный файл для mysql берётся тоже из этого репозитория), запуск mysqld сервера (на двух первых ядрах CPU), установка утилиты sysbench и подготовка проектных файлов для a-tune (как для сервера, так и для клиента), которые будут использоваться при поиске оптимальных параметров запуска mysqld.

Рассмотрим данные конфигурационные файлы:

Файл конфигурации сервера

 # cat /etc/atuned/tuning/mysql_sysbench_server.yaml

```
project: "mysql_sysbench"
maxiterations: 2048
startworkload: "`mysqld & ` & sleep 10"
stopworkload: "mysqladmin -S/var/lib/mysql/mysql.sock shutdown -uroot -p123456"
object :
  –
    name : "kernel.numa_balancing"
    info :
      desc : "Specifies whether to enable NUMA automatic balancing."
      get : "sysctl -n kernel.numa_balancing"
      set : "sysctl -w kernel.numa_balancing=$value"
      needrestart : "false"
      type : "discrete"
      options :
        – "0"
        – "1"
      dtype : "string"
  –
    name : "kernel.sched_autogroup_enabled"
    info :
        desc : "When enabled, the kernel creates task groups to optimize desktop program scheduling.
```

```
0: disabled
1: enabled"
    get : "sysctl -n kernel.sched_autogroup_enabled"
    set : "sysctl -w kernel.sched_autogroup_enabled=$value"
    needrestart : "false"
    type : "discrete"
    options :
      – "0"
      – "1"
    dtype : "string"
–
  name : "kernel.sched_wakeup_granularity_ns"
  info :
     desc : "This variable indicates the base of the minimum time that a process should run after it is woken up. The smaller the base, the higher the probability of preemption."
    get : "sysctl -n kernel.sched_wakeup_granularity_ns"
    set : "sysctl -w kernel.sched_wakeup_granularity_ns=$value"
    needrestart : "false"
    type : "discrete"
    scope :
      – 1000000
      – 100000000
    step : 1000000
    items :
    dtype : "int"
–
name : "kernel.sched_min_granularity_ns"
  info :
     desc : "Minimum running time of a process on the CPU. During this time, the kernel does not proactively select other processes for scheduling (in nanoseconds)."
    get : "sysctl -n kernel.sched_min_granularity_ns"
    set : "sysctl -w kernel.sched_min_granularity_ns=$value"
    needrestart : "false"
```

```
type: "discrete"
      scope:
        - 1000000
        - 100000000
      step: 1000000
      items:
      dtype: "int"
  -
name: "innodb_io_capacity"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_io_capacity'."
      get: "cat /etc/my.cnf | grep 'innodb_io_capacity' | awk -F '=' '{print $2}'"
      set: "sed -i 's/^innodb_io_capacity.*$/innodb_io_capacity=$value/g' /etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 100
        - 10000
      dtype: "int"
  -
    name: "innodb_write_io_threads"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_write_io_threads'."
      get: "cat /etc/my.cnf | grep 'innodb_write_io_threads' | awk -F '=' '{print $2}'"
            set : "sed -i 's/^innodb_write_io_threads.*$/innodb_write_io_threads=$value/g'
/etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 2
        - 200
```

```
dtype: "int"
  -
    name: "innodb_read_io_threads"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_read_io_threads'."
      get: "cat /etc/my.cnf | grep 'innodb_read_io_threads' | awk -F '=' '{print $2}'"
            set : "sed -i 's/^innodb_read_io_threads.*$/innodb_read_io_threads=$value/g'
/etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 2
        - 30
      dtype: "int"
  -
name: "innodb_spin_wait_delay"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_spin_wait_delay'."
      get: "cat /etc/my.cnf | grep 'innodb_spin_wait_delay' | awk -F '=' '{print $2}'"
            set : "sed -i 's/^innodb_spin_wait_delay.*$/innodb_spin_wait_delay=$value/g'
/etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 2
        - 30
      dtype: "int"
  -
name: "innodb_sync_spin_loops"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_sync_spin_loops'."
      get: "cat /etc/my.cnf | grep 'innodb_sync_spin_loops' | awk -F '=' '{print $2}'"
```

```
        set : "sed -i 's/^innodb_sync_spin_loops.*$/innodb_sync_spin_loops=$value/g'
/etc/my.cnf"
      needrestart : "true"
      type : "continuous"
      scope :
        – 10
        – 500
      dtype : "int"
  –
    name : "innodb_log_file_size"
    info :
      desc : "MySQL [mysqld] parameters 'innodb_log_file_size'."
       get : "cat /etc/my.cnf | grep 'innodb_log_file_size' | awk -F '=' '{print $2}' | awk -F 'M' '{print
$1}'"
      set : "sed -i 's/^innodb_log_file_size.*$/innodb_log_file_size=$valueM/g' /etc/my.cnf"
      needrestart : "true"
      type : "continuous"
      scope :
        – 10
        – 1024
      dtype : "int"
  –
  name : "innodb_log_files_in_group"
    info :
      desc : "MySQL [mysqld] parameters 'innodb_log_files_in_group'."
      get : "cat /etc/my.cnf | grep 'innodb_log_files_in_group' | awk -F '=' '{print $2}'"
        set : "sed -i 's/^innodb_log_files_in_group.*$/innodb_log_files_in_group=$value/g'
 /etc/my.cnf"
      needrestart : "true"
      type : "continuous"
  scope :
```

```
        – 1
        – 20
      dtype: "int"
  –
    name: "innodb_buffer_pool_instances"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_buffer_pool_instances'."
      get: "cat /etc/my.cnf | grep 'innodb_buffer_pool_instances' | awk -F '=' '{print $2}'"
      set: "sed -i
's/^innodb_buffer_pool_instances.*$/innodb_buffer_pool_instances=$value/g' /etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        – 1
        – 20
      dtype: "int"
  –
    name: "innodb_log_buffer_size"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_log_buffer_size'."
        get: "cat /etc/my.cnf | grep 'innodb_log_buffer_size' | awk -F '=' '{print $2}' | awk -F 'M'
'{print $1}'"
              set : "sed -i 's/^innodb_log_buffer_size.*$/innodb_log_buffer_size=$valueM/g'
/etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        – 8
        – 1024
    dtype: "int"
  –
```

```
name: "innodb_page_cleaners"
  info:
    desc: "MySQL [mysqld] parameters 'innodb_page_cleaners'."
    get: "cat /etc/my.cnf | grep 'innodb_page_cleaners' | awk -F '=' '{print $2}'"
    set: "sed -i 's/^innodb_page_cleaners.*$/innodb_page_cleaners=$value/g' /etc/my.cnf"
    needrestart: "true"
    type: "continuous"
    scope:
      - 1
      - 20
    dtype: "int"
  -
  name: "innodb_lru_scan_depth"
  info:
    desc: "MySQL [mysqld] parameters 'innodb_lru_scan_depth'."
    get: "cat /etc/my.cnf | grep 'innodb_lru_scan_depth' | awk -F '=' '{print $2}'"
    set : "sed -i 's/^innodb_lru_scan_depth.*$/innodb_lru_scan_depth=$value/g'
/etc/my.cnf"
    needrestart: "true"
    type: "continuous"
    scope:
      - 20
      - 2000
    dtype: "int"
  -
name: "innodb_thread_concurrency"
  info:
    desc: "MySQL [mysqld] parameters 'innodb_thread_concurrency'."
    get: "cat /etc/my.cnf | grep 'innodb_thread_concurrency' | awk -F '=' '{print $2}'"
    set : "sed -i 's/^innodb_thread_concurrency.*$/innodb_thread_concurrency=$value/g'
/etc/my.cnf"
```

```
needrestart: "true"
    type: "continuous"
    scope:
      - 0
      - 300
    dtype: "int"
  -
    name: "innodb_flush_log_at_trx_commit"
    info:
      desc: "MySQL [mysqld] parameters 'innodb_flush_log_at_trx_commit'."
      get: "cat /etc/my.cnf | grep 'innodb_flush_log_at_trx_commit' | awk -F '=' '{print $2}'"
      set: "sed -i 's/^innodb_flush_log_at_trx_commit.*$/innodb_flush_log_at_trx_commit=$value/g' /etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 0
        - 2
      dtype: "int"
  -
  name: "sync_binlog"
    info:
      desc: "MySQL [mysqld] parameters 'sync_binlog'."
      get: "cat /etc/my.cnf | grep 'sync_binlog' | awk -F '=' '{print $2}'"
      set: "sed -i 's/^sync_binlog.*$/sync_binlog=$value/g' /etc/my.cnf"
      needrestart: "true"
      type: "continuous"
      scope:
        - 0
        - 2
      dtype: "int"
```

В конфигурационном файле **/etc/atuned/tuning/mysql_sysbench_server.yaml** представлены команды запуска и остановки mysqld сервера, а также выбранные для тюнинга параметры конфигурации, у которых указаны верхние и нижние пределы значений (диапазон значений), которые A-Tune будет использовать для выбора при переборе параметров для запуска mysqld.

Файл конфигурации клиента:
 # cat mysql_sysbench_client.yaml

```
project: “mysql_sysbench”

engine : “gbrt”

iterations : 30

random_starts : 10



benchmark : “sh /root/A-
Tune/examples/tuning/mysql_sysbench/mysql_sysbench_benchmark.sh”

evaluations :

 –

  name: “QPS”

  info:

    get: “cat /root/A-
Tune/examples/tuning/mysql_sysbench/sysbench_oltp_read_write.log  | grep 'queries:' |
awk -F '(' '{print $2}' | awk -F ' ' '{print $1}'”

    type: “negative”

    weight: 100
```

Из представленного выше конфигурационного файла клиента видно, что оценка производительности идёт по параметру QPS, значение которого берётся из журнала работы **sysbench.**

Для запуска процесса оптимизации параметров настройки используем команду:

 # atune-adm tuning –project mysql_sysbench –detail mysql_sysbench_client.yaml

Эта команда будет работать продолжительное время и поэтому её лучше всего запустить в **screen** (или **tmux**, смотря кто что предпочитает).

По окончанию работы **atune-adm** напечатает в своем журнале работы те параметры настройки mysqld, при которых удалось достичь максимума QPS.

Пример вывода по результатам работы процесса оптимизации параметров:

The final optimization result is:
kernel.numa_balancing=0,kernel.sched_autogroup_enabled=1,kernel.sched_wakeup_granularity_ns=75000000,kernel.sched_min_granularity_ns=36000000,innodb_io_capacity=3184,innodb_write_io_threads=136,innodb_read_io_threads=11,innodb_spin_wait_delay=7,innodb_sync_spin_loops=174,innodb_log_file_size=781,innodb_log_files_in_group=6,innodb_buffer_pool_instances=5,innodb_log_buffer_size=500,innodb_page_cleaners=8,innodb_lru_scan_depth=706,innodb_thread_concurrency=249,innodb_flush_log_at_trx_commit=2,sync_binlog=0

The final evaluation value is: QPS=15031.05

Baseline Performance is: (QPS=12257.15)

Tuning Finished

3. Применение AI модели для поиска оптимальных параметров настройки приложения

Как мы говорили в начале статьи, в лабораторных условиях невозможно просчитать всех сценариев использования серверной системы и учесть все нюансы ее эксплуатации таким образом чтобы заранее подготовить и снабдить A-Tune профилями “на все возможные случаи жизни”. Как быть, если мы используем некое собственноручно написанное приложение или даже просто стороннее программное решение для которого нет написанного профиля? На помощь придет AI модуль A-Tune который позволит создать свой собственный уникальный профиль для использования в каждой конкретной ситуации и используемого приложения.

Допустим, наше приложение (под работу которого надо оптимизировать параметры ОС) называется testapp1 и оно установлено на сервер, где уже есть установленный A-Tune.

Для получения результата, нам следует выполнить следующий ряд действий:

Выполнить команду **a-tune**, которая создаёт новый профиль. При этом, для создания мы используем шаблонный файл профиля с пустыми значениями, как представлено на примере файла testapp1.conf ниже.

```
# cat testapp1.conf
[main]
# list its parent profile
[kernel_config]
# to change the kernel config
[bios]
# to change the bios config
[bootloader.grub2]
# to change the grub2 config
[sysfs]
# to change the /sys/* config
[systemctl]
# to change the system service status
[sysctl]
# to change the /proc/sys/* config
[script]
# the script extension of cpi
[ulimit]
# to change the resources limit of user
[schedule_policy]
# to change the schedule policy
[check]
# check the environment
[tip]
# the recommended optimization, which should be performed manually
```

Общий вид команды создания профиля:

```
atune-adm define [service_type] [application_name] [scenario_name] [profile_path]
```

Новый профиль создаем командой:

```
# atune-adm define testapp1_service testapp1 testapp1_scenario ./testapp1.conf
```

Теперь, если мы посмотрим список профилей доступных в A-Tune, то увидим новый, точно что нами созданный:

```
# atune-adm list | grep testapp1
| testapp1_service-testapp1-testapp1_scenario   | false    |
```

Заметим, что в данный момент профиль не является активным.

2. Следующим этапом будет сбор данных (то что в AI терминологии именуется Dataset) для обучения новой AI модели. Для этого, мы должны запустить наше приложение testapp1 и дать команду для сбора данных:

```
atune-adm collection –filename name –interval 5 –duration 120 –output_path /root/data
–disk sda –network enp1s0 -t testapp1_service-testapp1-testapp1_scenario
```

Где:

- duration – длительность (в секундах) сбора данных,
- sda – диск, на котором находится файловая система, которая используется приложением testapp1,
- enp1s0 – имя сетевого адаптера,
- testapp1_service-testapp1-testapp1_scenario – имя профиля, который мы создали ранее.

После окончания работы этой команды, в каталоге /root/data будут находится csv файлы с собранной статистикой.

3. Собранную статистику будем использовать для обучения новой модели

```
atune-adm train –data_path /root/data/ –output_file
/usr/libexec/atuned/analysis/models/new-model.m
```

4. И теперь, если мы дадим команду автоопределения профиля (с использованием созданной модели) под работу нашего приложения:

```
# atune-adm analysis –model /usr/libexec/atuned/analysis/models/new-model.m
```

то увидим, что A-Tune автоматически применил созданный нами в самом начале профиль, который работает согласно той модели, которую мы “натренировали” на собранных данных.

И в списке профилей, наш профиль уже является активным:

```
# atune-adm list | grep testapp1
| testapp1_service-testapp1-testapp1_scenario   | true     |
```

Таким образом, наличие инструмента A-Tune открывает новую страницу в процессе совершенствования инструментария оптимизации настройки сложных систем, максимально облегчая рабочий процесс системных администраторов, ответственных за управление ИТ инфраструктурой. В рамках нашего сообщества планируется дальнейшее совершенствование и развитие продукта, в частности для максимальной адаптации его к текущим реалиям российского рынка серверных ОС. Планируется доработка библиотеки готовых профилей использования и подробное документирование процесса использования данного инструмента в связке с широко известными и востребованными на российском корпоративном рынке СПО решениями, а также с решениями российских разработчиков, работающими на дистрибутиве OpenScaler.