АО "НТЦ ИТ РОСА"

ОПЕРАЦИОННАЯ СИСТЕМА РОСА "ХРОМ" 12

(Несертифицированная версия)

Руководство администратора

РСЮК.10501-01 32 01

Листов 488

2025

АННОТАЦИЯ

Данный документ является руководством администратора операционной системы РОСА "ХРОМ" версии 12 РСЮК.10501-01 (далее – ОС, Система) в вариантах "рабочая станция" (далее – ОС рабочей станции) и "сервер" (далее – серверная ОС).

ОС зарегистрирована в реестре российского ПО от 05.09.2016 №1607.

Предполагается, что читатель является системным администратором ОС Linux с опытом работы в интерфейсе командной строки. Также рекомендуется ознакомиться с принципами работы Системы в графическом интерфейсе пользователя, которые описаны в документе "ОС РОСА "ХРОМ". Руководство пользователя. РСЮК.10501-01 34 01".

В настоящем руководстве приведен состав ОС, инструкции по нескольким типам установки ОС (локальная, установка с использованием средств маскирования информации, установка в виртуальную машину, автоматизированная установка), общие сведения о работе в ОС, ее настройке и организации работы пользователей в ней.

Перед установкой, настройкой и эксплуатацией ОС рекомендуется внимательно ознакомиться с настоящим руководством.

Для разработки документа использованы ссылки на следующие стандарты:

– ГОСТ Р 2.105-2019 "Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам";

– ГОСТ 2.601 "Единая система программной документации. Виды программных документов";

– ГОСТ 19.101-77 "Единая система программной документации. Виды программ и программных документов";

– ГОСТ 19.105-78 "Единая система программной документации. Общие требования к программным документам";

– ГОСТ 19.503-79 "Единая система программной документации. Руководство системного программиста".

Настоящий документ подготовлен в соответствии с технологической инструкцией "РОСА. Регламент формирования документации к программным продуктам" (индекс РСЮК.11001-02 90 02).

СОДЕРЖАНИЕ

1 Общие сведения 11
1.1 Общее описание 11
1.2 Основные характеристики 11
1.2.1 Аппаратная поддержка 11
1.2.2 Программная платформа 11
1.2.3 Программное обеспечение 11
1.2.4 Сети и развертывание 12
1.2.5 Защита информации 13
1.2.6 Состав дистрибутива 13
1.2.7 Доступные в репозитории компоненты 14
1.3 Приемка ОС 15
1.4 Требования к техническим средствам 16
1.5 Состав ОС 16
2 Установка Системы 17
2.1 Безопасная установка и эксплуатация ОС 17
2.1.1 Условия безопасной эксплуатации 17
2.1.2 Доверенная загрузка 17
2.1.3 Защита информации 17
2.1.4 Безопасная конфигурация ОС и ее целостность 18
2.1.5 Пользователи и администраторы 18
2.1.6 Настройка точного времени 18
2.2 Локальная установка 19
2.2.1 Подготовка к установке 19
2.2.2 Установка ОС на рабочую станцию 19
2.2.3 Установка серверной версии ОС 26
2.2.4 Парольная защита загрузчика ОС 26
2.3 Установка с использованием средств маскирования информации 27
2.3.1 Общие вопросы 27
2.3.2 Разметка носителя с маскированием информации 27
2.3.3 Поддержка tpm2 для автоматического ввода паролей 29
2.4 Установка на виртуальную машину 29
2.5 Автоматизированное развертывание 29
2.5.1 Введение в kickstart-сценарии 29
2.5.2 Типовой сценарий автоматизированной установки 30
2.5.3 Дополнительные действия в kickstart-сценариях 33
2.6 Сетевая загрузка с возможностью установки 34
2.6.1 Технология Intel PXE 34
2.6.2 Начальная настройка сервера для работы PXE 35
2.6.3 Настройка сервера DHCP (BOOTP) 35

2.6.4 Настройка веб-сервера....36
2.6.5 Настройка сервера TFTP....37
2.6.6 Настройка сети в LiveCD....38
2.7 Графическая установка с использованием VNC....39
2.8 OEM-установка....40
2.9 Установщик iso2img....42
2.10 Подготовительные процедуры после установки....42
3 Общие сведения о работе в Системе....44
3.1 Интерфейсы Системы....44
3.1.1 Общие сведения о работе в консоли....44
3.2 Администраторы и пользователи....47
3.2.1 Общая информация....47
3.2.2 Списки пользователей....49
3.2.3 Использование sudo и привилегии....49
3.2.4 Управление учетными записями пользователей....53
3.2.5 Принудительное завершение сеанса пользователя....56
3.2.6 Восстановление забытого пароля пользователя....57
3.2.7 Группы пользователей....59
3.3 Команды....62
3.3.1 Текстовые редакторы....64
3.3.2 Выполнение команд....68
3.3.3 Фоновые процессы....69
3.3.4 Выполнение нескольких команд....70
3.3.5 Потоки и перенаправление....72
3.3.6 Поиск и фильтрация информации....72
3.3.7 Переменные окружения....80
3.3.8 Справочные ресурсы по командам....81
3.4 Системный менеджер systemd....84
3.4.1 Управление службами....84
3.4.2 Unit-файлы....85
3.4.3 Состояние Системы....92
3.4.4 systemd-resolved....93
3.5 Файловые системы....95
3.5.1 Поддерживаемые файловые системы....96
3.5.2 Организация каталогов в Системе....98
3.5.3 Монтирование....100
3.5.4 Размонтирование....103
3.5.5 Файловая система BTRFS....104
3.5.6 Файловая система ZFS....119
3.5.7 Файловая система GFS2....128
3.5.8 Указатели....129
3.5.9 Безопасное удаление файлов....131

3.6 Установка и удаление программ (репозитории) 133
3.6.1 Использование rpm 134
3.6.2 Файловый менеджер dnf 135
3.6.3 Установка программ из сторонних источников 136
3.6.4 Использование Notamock для сборки пакетов 137
3.6.5 Архивный репозиторий пакетов repo-archive 141
3.7 Управление жесткими дисками 142
3.7.1 Fdisk 143
3.7.2 Gdisk 146
3.7.3 Диспетчер разделов KDE 150
3.7.4 GParted 153
3.7.5 Управление дисковыми квотами 155
3.7.6 Блокирование файлов (fileprotect) 157
3.8 Режимы работы Системы 159
3.8.1 Режим восстановления 159
3.8.2 Аварийный режим 159
3.8.3 Синхронизация с репозиториями 160
3.8.4 Восстановление функционирования ОС 160
3.8.5 Восстановление загрузчика GRUB2 (MBR и EFI) 161
3.8.6 Восстановление Системы из среды chroot 163
3.9 Перезагрузка и остановка Системы 165
3.9.1 Перезагрузка 165
3.9.2 Остановка 166
3.9.3 Использование команды shutdown 166
3.10 Обновление Системы 168
3.10.1 Общие сведения 168
3.10.2 Индикация обновлений 168
3.10.3 Порядок получения обновлений 169
3.11 Сохранение вывода консоли в файл 170
3.12 Запись экрана из консоли 172
3.12.1 Перекодирование видео 173
3.13 Процессы 174
3.13.1 Жизненный цикл процесса 174
3.13.2 Приоритезация процессов 176
3.13.3 Получение информации о процессе 177
3.13.4 Управление процессами 180
3.14 Отладка и диагностика 187
3.14.1 Консольный отладчик GDB 187
3.14.2 Диагностика и отладка процессов с помощью утилиты strace 189
3.14.3 Мониторинг и анализ состояния системы через псевдо-файловую систему /proc (procfs) 190

4 Управление пользователями ... 192
4.1 Управление учётными данными пользователей ... 192
4.1.1 Настройка основных параметров пользователей в /etc/passwd/ ... 192
4.1.2 Управление паролями в/etc/shadow/ ... 193
4.2 Управление механизмами аутентификации и авторизации через модули PAM ... 195
4.2.1 Механизм работы PAM ... 196
4.2.2 Конфигурация распространённых модулей PAM ... 198
4.2.3 Блокирование неактивных учётных записей ... 204
4.2.4 Отладка и проверка работы PAM ... 205
4.3 Настройка двухфакторной аутентификации ... 207
4.3.1 Подготовка рабочих станций ... 208
4.3.2 Подготовка токена Рутокен ... 208
4.3.3 Извлечение и перенос публичного ключа ... 209
4.3.4 Подготовка целевой станции ... 209
4.3.5 Список ключей пользователя ... 210
4.3.6 Создание общего файла настроек PAM ... 211
4.3.7 Расширенная настройка ... 212
4.4 Управление пользователями в локальной сети с FreeIPA ... 217
4.4.1 Настройка сервера домена IPA ... 217
4.4.2 Настройка клиента домена IPA ... 220
4.4.3 Вход в домен ... 221
5 Сетевые подключения ... 222
5.1 Способы настройки сетевых соединений ... 222
5.1.1 NetworkManager ... 222
5.1.2 Использование nmtui ... 223
5.1.3 Использование nmcli ... 224
5.1.4 Использование ifconfig и ip ... 225
5.1.5 Виртуальный сетевой интерфейс Loopback ... 227
5.1.6 Сетевой интерфейс ens* ... 228
5.2 Настройка статических соединений ... 229
5.2.1 Настройка статического IP-адреса ... 231
5.3 Настройка динамических соединений ... 231
5.3.1 Настройка динамических соединений (кроме DNS) ... 231
5.4 Переадресация ... 232
5.4.1 Включение переадресации IPv4 ... 233
5.4.2 Включение переадресации IPv6 ... 233
5.4.3 Настройка правил переадресации с использованием nftables ... 234
5.5 Мониторинг и диагностика сети ... 235

5.6 Межсетевой экран Nftables 236
5.6.1 Общие сведения о Nftables 236
5.6.2 Установка и запуск Nftables 238
5.6.3 Работа с Nftables 238
5.7 Синхронизация времени с использованием Chrony 240
5.7.1 Настройка Chrony в качестве сервера времени 242
6 Настройка и управление сетевыми сервисами 243
6.1 Сервер DHCP 243
6.1.1 Установка 243
6.1.2 Настройка 243
6.1.3 Присвоение адреса 245
6.1.4 Настройка клиента 246
6.2 Сервер DNS 246
6.2.1 Установка 246
6.2.2 Настройка 247
6.2.3 Переадресация 248
6.3 Сервер Samba 249
6.3.1 Настройка сети 249
6.3.2 Запуск контроллера домена 251
6.3.3 Перенаправление 253
6.3.4 Использование внешнего DNS-сервера BIND 255
6.3.5 Подключение клиента ROSA к домену 256
6.4 Веб-серверы 261
6.4.1 Apache 261
6.4.2 Веб-сервер Nginx 265
6.4.3 Веб-сервер Angie 283
6.5 Веб-интерфейс управления Cockpit 297
6.6 Сервер журналирования 299
6.6.1 journald 299
6.6.2 rsyslog 304
6.7 Сервер печати CUPS 310
6.7.1 Добавление и настройка принтеров 311
6.8 Почтовый сервер OpenSMTPD 314
6.8.1 Настройка OpenSMTPD 315
6.8.2 Проверка работы OpenSMTPD 316
6.8.3 Интеграция OpenSMTPD с Dovecot 318
6.8.4 Дополнительные рекомендации по безопасности и интеграции 319
6.9 Почтовый сервер Postfix 322
6.9.1 Установка и начальная подготовка 322

6.9.2 Основные параметры конфигурации 323
6.9.3 Настройка исходящей почты (SMTP-клиент) 325
6.9.4 Настройка входящей почты 326
6.9.5 Настройка безопасности 327
6.9.6 Управление очередями сообщений 328
6.9.7 Проверка работы и диагностика 329
6.9.8 Примеры типовых конфигураций 330

7 Безопасность 332

7.1 Контроль доступа к файлам 332

7.1.1 Общие сведения о правах доступа 332
7.1.2 Смена владельца файла или каталога 334
7.1.3 Изменение прав доступа 334
7.1.4 Изменение атрибутов файлов 336
7.1.5 Расширенные списки контроля доступа (ACL) 340

7.2 Резервное копирование данных 346

7.2.1 Планирование резервного копирования и восстановления данных 347
7.2.2 Создание резервных копий 348
7.2.3 Восстановление данных из резервной копии 352
7.2.4 Архивирование 352
7.2.5 Сжатие 356

7.3 Контроль целостности Системы 361

7.3.1 Контроль целостности с помощью AIDE 361
7.3.2 Контроль целостности на основе Afick 365

7.4 Формирование замкнутой программной среды 369

7.4.1 Общие принципы 369
7.4.2 Создание ключей 370
7.4.3 Создание политики IMA 371
7.4.4 Подписывание файлов 372
7.4.5 Подготовка файловой системы 374
7.4.6 Добавление в образ начальной загрузки 374
7.4.7 Пробный запуск подписанной Системы 374
7.4.8 Рабочий запуск и проверка работы 375

7.5 Конфигурация системы для защиты информации 376

7.5.1 Общее описание 376
7.5.2 Настройки службы SSH 376
7.5.3 Отключение файловых систем без атрибутов безопасности 376
7.5.4 Отключение редких сетевых протоколов 376
7.5.5 Отключение дампов оперативной памяти 377
7.5.6 Настройка переменных ядра 377
7.5.7 Настройки аудита 377
7.5.8 Отключение настроек блокировки экрана 377
7.5.9 Настройка автоблокировки консольного режима 378

7.6 Аудит 378
7.6.1 Общие сведения о auditd 378
7.6.2 Управление службой аудита 379
7.6.3 Настройка конфигурации аудита 379
7.6.4 Настройка правил аудита 379
7.6.5 Выборка из логов аудита 381
7.6.6 Примеры работы со службой аудита 382
7.7 Отказоустойчивый кластер 386
7.7.1 Общие сведения 386
7.7.2 Развёртывание кластера высокой доступности 388
7.7.3 Управление ресурсами и политиками в кластере 389
7.7.4 Проверка работоспособности кластера 391
7.7.5 Мониторинг и управление отказами 392
7.7.6 Управление ресурсами и диагностика 392
8 Удаленный доступ к серверу 394
8.1 Доступ по OpenSSH 394
8.1.1 Общие сведения об OpenSSH 394
8.1.2 Настройка SSH 394
8.1.3 Пары ключей 396
8.1.4 Процессы, запущенные пользователями, сеанс которых завершен 398
8.2 Терминальный доступ с использованием XRDP 405
8.2.1 Настройка локальных пользователей 406
8.2.2 Настройка доменных пользователей 407
8.2.3 Подключение с клиента 408
9 Автоматизация инфраструктуры 409
9.1 Написание bash-скриптов 409
9.1.1 Команды 409
9.1.2 Сценарии 410
9.1.3 Ввод и вывод данных 412
9.1.4 Функции и аргументы 413
9.1.5 Области видимости и потоки 415
9.1.6 Операторы 416
9.1.7 Циклы и конструкции 419
9.1.8 Арифметические операции и массивы 421
9.2 Ansible 423
9.2.1 Установка и подключение хостов 423
9.2.2 Проверка подключения 424
9.2.3 Описание сценариев Ansible 425
9.2.4 Создание сценариев 426
9.2.5 Выполнение одиночных команд 427
10 Контейнеризация 428

10.1 Docker 428
10.2 Podman 430
10.2.1 Общие сведения 430
10.2.2 Установка и базовая настройка 432
10.2.3 Работа с контейнерами Podman 434
10.2.4 Интеграция Podman с systemd 437
10.3 Kubernetes 439
10.3.1 Общие сведения 439
10.3.2 Установка kubernates master-node 440
10.3.3 Установка kubernates worker-nodes (pods-контейнеры) 441
11 Сбор отчетов о производительности 444
11.1 Общие сведения о Zabbix 444
11.2 Установка и настройка 445
11.2.1 Установка компонентов Zabbix 445
11.2.2 Настройка веб-интерфейса 446
11.2.3 Установка Zabbix Agent на целевых машинах 446
11.2.4 Завершение настройки 447
12 Решение прикладных задач 448
12.1 Системы управления базами данных 448
12.1.1 Доступные СУБД 448
12.1.2 PostgreSQL 448
12.1.3 MySQL 449
12.1.4 SQLITE 450
12.2 Локальная виртуализация с использованием qemoo 451
12.2.1 Запуск виртуальных машин с помощью qemoo 451
12.2.2 Установка ОС на виртуальный диск 453
12.2.3 Сетевые режимы и проброс устройств 453
12.2.4 Работа в демон-режиме (SPICE) 454
12.2.5 Конфигурация и приоритет параметров 456
13 Создание собственного репозитория пакетов 458
13.1 Зеркалирование репозиториев 458
13.2 Создание собственного репозитория 459
13.3 Подключение локального репозитория 460
13.4 Доступ к репозиторию по сети 460
13.4.1 Доступ по HTTP 460
13.4.2 Доступ по NFS 462
Приложение А. Список доступных консольных команд 464
Перечень терминов и сокращений 487

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Общее описание

ОС РОСА "ХРОМ" предназначена для оснащения рабочих мест пользователей в корпоративной и государственной среде, предоставляя пользователям современный и интуитивно понятный интерфейс.

Для ОС доступны образы с окружением рабочего стола "KDE Plasma 5".

Версии ОС представлены в исполнениях "рабочая станция" и "сервер", которые позволяют использовать Систему в качестве базовой платформы по выполнению рутинных задач разной степени сложности.

1.2 Основные характеристики

1.2.1 Аппаратная поддержка

Образ продукта собран на новом LTS-ядре Linux 6.12. Ядра веток 6.6, 6.1, 5.10 и 5.15 доступны в репозиториях.

Поддержка множества Wi-Fi-карт осуществляется с помощью модулей ядра, а также bluetooth-модулей – в большинстве случаев "из коробки".

Обеспечена поддержка новейших видеокарт NVIDIA: репозитории содержат драйверы Nvidia 340, 390, 470, 510, 515, 520, 525, 550, 580 в виде модулей ядра с автоустановкой посредством фирменной консольной утилиты kroko-cli.

Система содержит обновлённые библиотеки трёхмерной графики MESA.

1.2.2 Программная платформа

Программная платформа 2021.1 построена на независимой от других дистрибутивов пакетной базе.

Платформа реализована на основе собственной среда сборки abf.io, в которой пользователи могут сами собирать нужные им пакеты и создавать собственные репозитории.

В платформе используются системные программные продукты: glibc 2.33, gcc 11.2, Java версий 1.8, 11, 17.

1.2.3 Программное обеспечение

В качестве прикладного ПО в составе ОС используются:

– свежие пакеты программ, в большинстве своем соответствующие стабильной ветке Upstream;

– рабочая оболочка на основе KDE Plasma 5.27 с собственным вариантом оформления и поддержкой темной темы, в том числе добавлен новый алгоритм позиционирования иконок на рабочем столе из версии Plasma 6.3;

– фирменный индикатор обновлений Системы, автопереключение на резервные "зеркала" при недоступности основных;

– в Систему добавлены российские корневые сертификаты ТЦИ.

В состав ОС включены следующие категории дополнительных системных приложений:

– архиваторы и утилиты сжатия;

– средства управления пакетами и обновлениями (DNF/RPM);

– приложения мониторинга, журналирования и диагностики;

– приложения для работы с файлами и файловыми системами;

– средства настройки системы и учётных записей (локализация, дата/время, службы systemd);

– настройка параметров загрузки и восстановления (GRUB2, целевые состояния systemd);

– настройка оборудования (графика, звук, печать, Bluetooth и др.);

– утилиты настройки сети (NetworkManager);

– межсетевой экран и NAT на базе nftables;

– средства удалённого доступа и администрирования (OpenSSH, Xrdp);

– инструменты резервного копирования и переноса данных;

– средства работы с образами и установочными носителями (создание/запись ISO).

1.2.4 Сети и развертывание

В Системе реализованы следующие механизмы сетевой конфигурации и автоматизированного развертывания:

– автоматизированная установка по сценариям kickstart: задание разметки дисков, набора пакетов, сетевых параметров, pre-/post-скриптов; поддерживается самостоятельное задание учётных данных при первом входе в Систему;

– установка по сети с использованием PXE/iPXE, загрузка по HTTP/HTTPS (включая UEFI), подключение установочных репозиториев по HTTPS с проверкой криптографических подписей пакетов;

– управление сетевыми подключениями на базе NetworkManager (утилиты nmcli/nmtui): проводные и беспроводные сети, статическая и динамическая (DHCPv4/DHCPv6) конфигурация, DNS и маршрутизация, поддержка IPv4/IPv6

(dual-stack), VLAN, мостирование (bridge), агрегирование (bonding), 802.1X, VPN (например, OpenVPN, IPsec, WireGuard), настройка прокси;

– синхронизация времени по NTP (chrony) с поддержкой настройки источников времени и корректной работы в доменной среде;

– межсетевой экран на базе nftables для фильтрации трафика и NAT на уровне узла; интеграция с systemd (управление сервисами правил, журналирование и контроль состояния);

– присоединение к доменным инфраструктурам: IPA и Samba/Active Directory (через realmd/SSSD, Kerberos/LDAP), централизованная аутентификация, применение политик доступа, автоматическое создание домашних каталогов пользователей;

– использование репозиториев программного обеспечения по HTTPS, возможность организации локальных зеркал и кэширующих прокси для массовых развертываний; управление установками и обновлениями через DNF при низкоуровневом учёте пакетов в RPM;

– средства постустановочной конфигурации для тиражирования настроек (скрипты оболочки, профили systemd, при необходимости — инструменты оркестрации, такие как Ansible).

1.2.5 Защита информации

В части защиты информации в ОС поддерживается:

– запутывающее кодирование по ГОСТ в браузерах и OpenSSL;

– технология подписей исполняемых файлов IMA;

– установка и загрузка с шифрованных разделов без ввода пароля при загрузке (технология tpm2).

1.2.6 Состав дистрибутива

В состав образа сервера входят следующие компоненты:

– графический, консольный и автоматизированный установщики;

– ядро Linux 6.12;

– системный менеджер systemd;

– сервер SSH openssh-server;

– служба синхронизация времени chrony;

– служба автоматической или ручной настройки сети NetworkManager;

В состав образа рабочей станции так же входит графическая оболочка на основе KDE Plasma5 и комплект прикладных программ.

1.2.7 Доступные в репозитории компоненты

Другие компоненты можно установить из обширного официального репозитория РОСА, например:

– служба каталогов FreeIPA (пакет ipa-server);
– служба каталогов Samba AD;
– DNS-сервер bind;
– база данных MariaDB (MySQL) (пакет mariadb);
– база данных PostgreSQL (пакет postgresql, либо postgresql14, либо postgresql15);
– веб- и прокси-сервер apache (с поддержкой пакета apache-base);
– веб- и прокси-сервер nginx и различные модули для него (пакеты nginx-module-*);
– система мониторинга Zabbix (пакет zabbix5.0), Nagios, Monitorix, Monit и т.д.;
– виртуализация (qemu+libvirt);
– российское ответвление nginx – angie и те же самые модули, что и для nginx:

- angie-module-perl;
- angie-module-rtmp;
- angie-module-upload;
- angie-module-http_auth_pam;
- angie-module-ndk;
- angie-module-array-var;
- angie-module-aws-auth;
- angie-module-brotli;
- angie-module-spnego-auth;

– и многие другие.

Репозиторий ОС "сервер" является общим с репозиторием ОС "рабочая станция".

Также можно установить различные графические оболочки для ОС, однако рекомендуется использовать Систему с имеющейся в образе оболочкой KDE Plasma5. В остальных случаях разработчик не может гарантировать корректность работы ОС. Возможные варианты с графическими средами приведены ниже (таблица 1).

Таблица 1 – Графические оболочки

Графическая оболочка	Пакет для установки
XFCE	task-xfce для XFCE с настройками ОС или task-xfce-minimal для XFCE без доп. настроек
KDE Plasma5	task-plasma5 или task-iso-plasma5
GNOME	task-iso-gnome для установки GNOME с настройками, дополнениями и другим ПО, как в образе с GNOME; для установки "ванильного" GNOME: gdm gnome-session-x11-session gnome-session-wayland-session gnome-classic-session
LXQt	task-iso-lxqt
MATE	task-mate
Lumina	task-lumina
icewm	icewm или task-iso-icewm

Дополнительно необходимо установить пакет task-x11.

Примечание – Проверить целостность образов можно командой:

```
checkisomd5 ИмяОбраза
```

Также можно проверить целостность образов, выбрав при начальной загрузке Системы с инсталляционного носителя режим "Проверка целостности" в меню загрузчика "Диагностика".

1.3 Приемка ОС

Для приемки изделия необходимо:

а) проверить комплектность изделия; в комплект должны входить:

– оптические диски с ОС и комплектом документации согласно выбранному варианту поставки;

– лицензионное соглашение.

б) проверить контрольные суммы установочных дистрибутивов с Системой в соответствии с КС, указанной в файле sha1, который поставляется вместе с образом ОС.

в) приемка изделия считается успешной, если все указанные требования выполнены.

1.4 Требования к техническим средствам

Для корректной установки и функционирования ОС необходима следующая минимальная конфигурация оборудования (ПК):

– процессор с архитектурой x86_64 или AArch64, Байкал;
– оперативная память – 2 Гб (4 Гб – при загрузке ОС по сети);
– дисковое пространство – 25 Гб;
– VGA-адаптер и монитор с поддержкой разрешения 1024×768 пикс. (24 бит);
– устройство чтения оптических дисков или возможность загрузки с USB;
– клавиатура;
– компьютерная мышь.

1.5 Состав ОС

ОС включает в себя следующие основные компоненты:

– ядро и системные библиотеки;
– системные сервисы и программы;
– пользовательские программы.

Ядро и системные библиотеки обеспечивают взаимодействие ОС с аппаратной частью и выделение ресурсов для всех других компонентов.

Системные сервисы и программы дают возможность настройки и реконфигурации ОС в зависимости от задач пользователя и обслуживание повседневной работы.

Пользовательские программы предназначены для решения задач конечного пользователя.

2 УСТАНОВКА СИСТЕМЫ

2.1 Безопасная установка и эксплуатация ОС

2.1.1 Условия безопасной эксплуатации

Для безопасной эксплуатации ОС должна быть установлена на совместимую аппаратную часть, удовлетворяющую требованиям п.1.4.

Организационными или техническими средствами (с помощью пломб или замков) должна быть обеспечена защита физической целостности корпуса компьютера с установленной ОС от изменения аппаратной конфигурации.

Установку, настройку и обслуживание ОС должен проводить обладающий необходимыми компетенциями администратор ОС в соответствии с настоящим Руководством.

Эксплуатацию ОС должны выполнять обладающие необходимыми компетенциями пользователи в соответствии с документом "ОС РОСА "ХРОМ". Руководство пользователя. РСЮК.10501-01 34 01".

2.1.2 Доверенная загрузка

Для безопасной работы в ОС должна быть реализована её доверенная загрузка, обеспечиваемая внешними средствами АПМДЗ/СДЗ и/или организационными мерами, а именно:

– защитой BIOS паролем от изменения;

– отключением в BIOS загрузки с внешних носителей (DVD, подключаемых накопителей) согласно документации к аппаратуре после установки;

– защитой паролем режима изменения параметров загрузчика GRUB (п.2.2.4).

2.1.3 Защита информации

Администратор должен обеспечить механизмы резервного копирования и необходимый объем носителей для защищенного хранения резервных копий данных пользователей согласно инструкциям, описанным в п. 7.2.

При использовании парольной защиты для идентификации во всех случаях администратор должен использовать пароли достаточной степени сложности. При использовании небезопасных паролей ОС предупреждает об этом; при получении такого предупреждения нужно изменить пароль на более сложный.

Рекомендуется произвести установку ОС с использованием механизма маскирования информации на носителе согласно п.2.3.

Для предотвращения утечки информации с помощью записи её на съемные внешние накопители (флеш-накопитель, внешние HDD, DWD-RW и др.) должен быть организован процесс контроля их использования. Это возможно сделать как чисто организационными мерами, так и аппаратным отключением возможности подключения внешних накопителей на пользовательских компьютерах (с помощью настроек BIOS).

Для предотвращения несанкционированного доступа к информации в момент, когда авторизованный пользователь ОС делает перерыв в работе, он должен блокировать экран нажатием сочетания клавиш `META+L` на клавиатуре в графическом режиме или же командой `vlock`, если он работает в консольном текстовом режиме.

2.1.4 Безопасная конфигурация ОС и ее целостность

ОС также может быть сконфигурирована для безопасной работы с помощью файлов конфигурации, описанных в п.7.5. Необходимо контролировать с помощью программы AIDE целостность Системы путем создания базы данных после инсталляции, как показано в п.7.3, и периодического осуществления контроля.

2.1.5 Пользователи и администраторы

Первый созданный инсталлятором пользователь при установке по умолчанию обладает доступом к правам администратора через команду `sudo`, т. е. является администратором ОС. Эту учетную запись необходимо применять только для настройки ОС, пользователи должны работать под отдельной учетной записью с пользовательскими правами. Подробнее об этом см. п. 3.2.

2.1.6 Настройка точного времени

Для правильного ведения журналов событий и аудита в компьютере и ОС должно быть установлено актуальное системное время. Администратору необходимо контролировать установку правильного часового пояса и сохранение этого времени при выключении компьютера (поддерживать актуальность автономного питания BIOS с помощью встроенной батареи).

ОС сконфигурирована так, что время автоматически синхронизируется по сети Интернет с публичными серверами. Доступ к этой настройке имеют администраторы ОС, пользователь менять время по умолчанию не может.

2.2 Локальная установка

2.2.1 Подготовка к установке

Перед началом установки ОС рекомендуется выполнить следующие действия:

а) убедиться, что аппаратная конфигурация ПК удовлетворяет минимальным аппаратным требованиям, указанным в п.1.4;

б) убедиться в подлинности DVD с дистрибутивом ОС (п.1.3);

в) скопировать с DVD с дистрибутивом ОС образ на USB-носитель (если установка будет производиться с USB-носителя);

г) провести временную настройку BIOS/UEFI компьютера для обеспечения возможности загрузки с выбранного носителя, используя документацию на оборудование. В большинстве случаев выбор вариантов загрузки становится доступен при удержании клавиш `Del`, `F12` или `F2` при начале загрузки компьютера.

Для обеспечения доверенной загрузки ОС рекомендуется после успешной инсталляции режим загрузки с внешнего накопителя отключить.

2.2.2 Установка ОС на рабочую станцию

Выбор позиций меню при установке осуществляется стрелками на клавиатуре и нажатием клавиши `Enter` или же выбором элемента меню курсором и нажатием левой кнопки мыши.

Для того чтобы провести процедуру установки ОС, необходимо выполнить следующие действия:

а) вставить в устройство чтения DVD загрузочный диск изделия или USB-носитель с дистрибутивом ОС до того, как ПК начнет обращение к загрузочным дискам. Если обращение уже было произведено, и установка изделия не началась, необходимо перезагрузить ПК.

б) выбрать язык, на котором будет производиться установка.

в) выбрать вариант установки "Запуск и установка ROSA Linux (live)" (рекомендуется) или же режим "Установка ROSA Linux", при котором установка идет без предварительной загрузки ОС (рисунок 1). Такой режим применим для специальных случаев, например, при недостатке оперативной памяти или при необходимости ускорения установки.

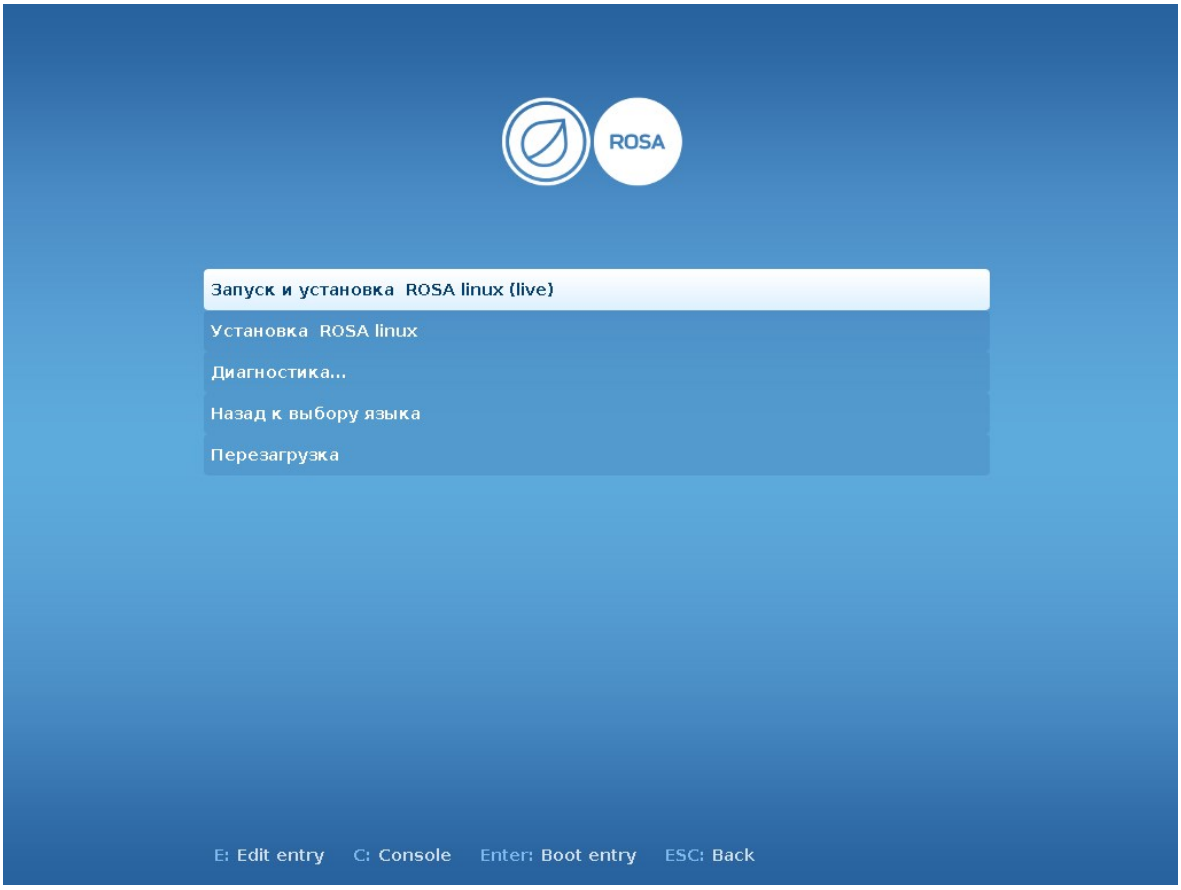


Рисунок 1 – Выбор варианта установки

г) При проблемах с установкой может быть полезным пункт "Диагностика", в подменю которого доступны варианты с проверкой контрольной суммы образа перед установкой, проверкой оперативной памяти ПК или установкой в режиме упрощенной графики либо в текстовом режиме. Перед обращением в техническую поддержку при каких-либо проблемах настоятельно рекомендуется как минимум провести диагностику оперативной памяти (полный цикл) и попробовать загрузить установочный образ с предварительным тестированием контрольной суммы, что позволит исключить ошибки аппаратного обеспечения. При наличии сбоев оперативной памяти необходимо заменить ее на исправную, при сбое контрольной суммы образа следует использовать другой носитель или другой USB-порт. Пример меню "Диагностика" приведен ниже (рисунок 2) (конкретный внутренний номер образа зависит от версии и может понадобиться при обращении в техническую поддержку).

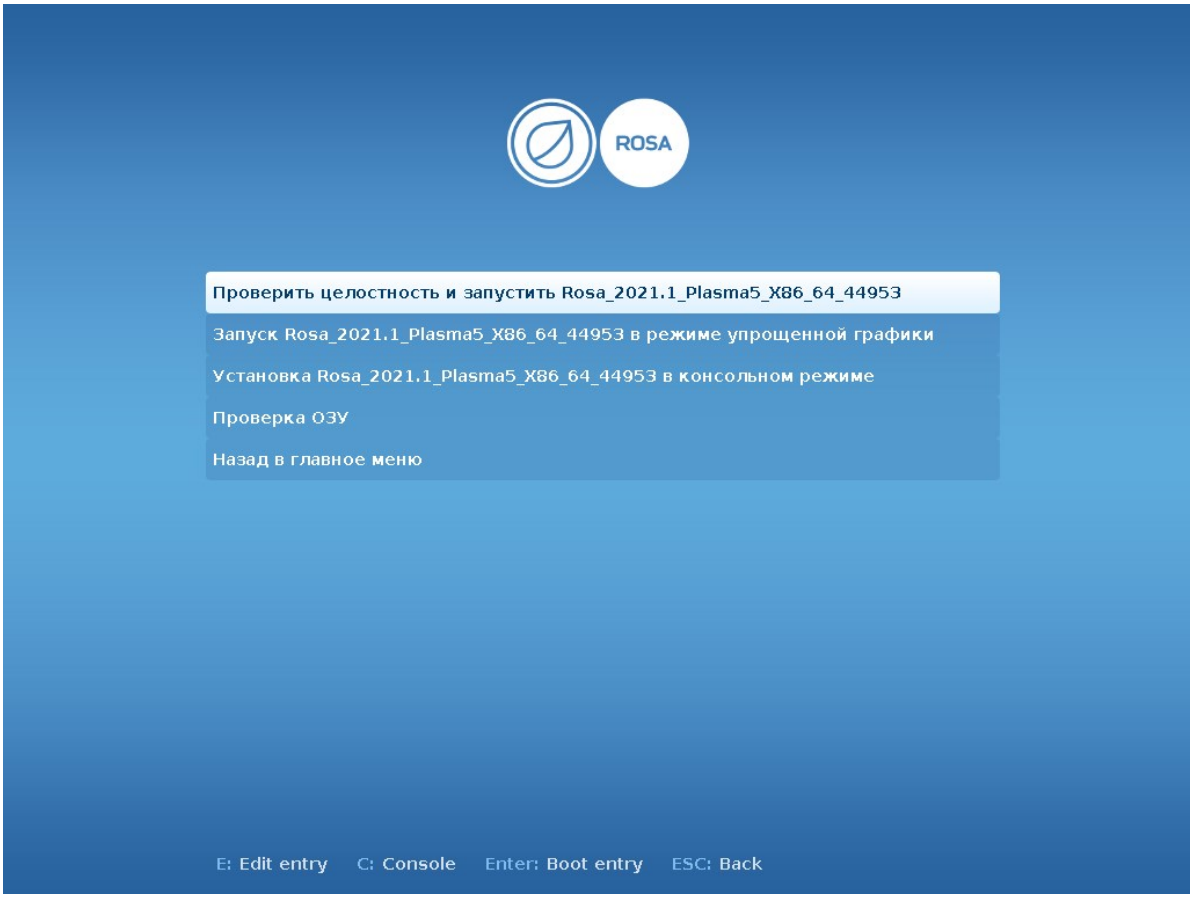


Рисунок 2 – Меню "Диагностика"

д) После загрузки в live-режиме появляется возможность полноценной проверки работы ОС на конкретном ПК с возможностью установки с помощью ярлыка инсталлятора на рабочем столе (рисунок 3). Перед установкой рекомендуется подключить ОС к сети (используя пиктограмму на панели задач), тогда часть параметров будет заполнена автоматически.

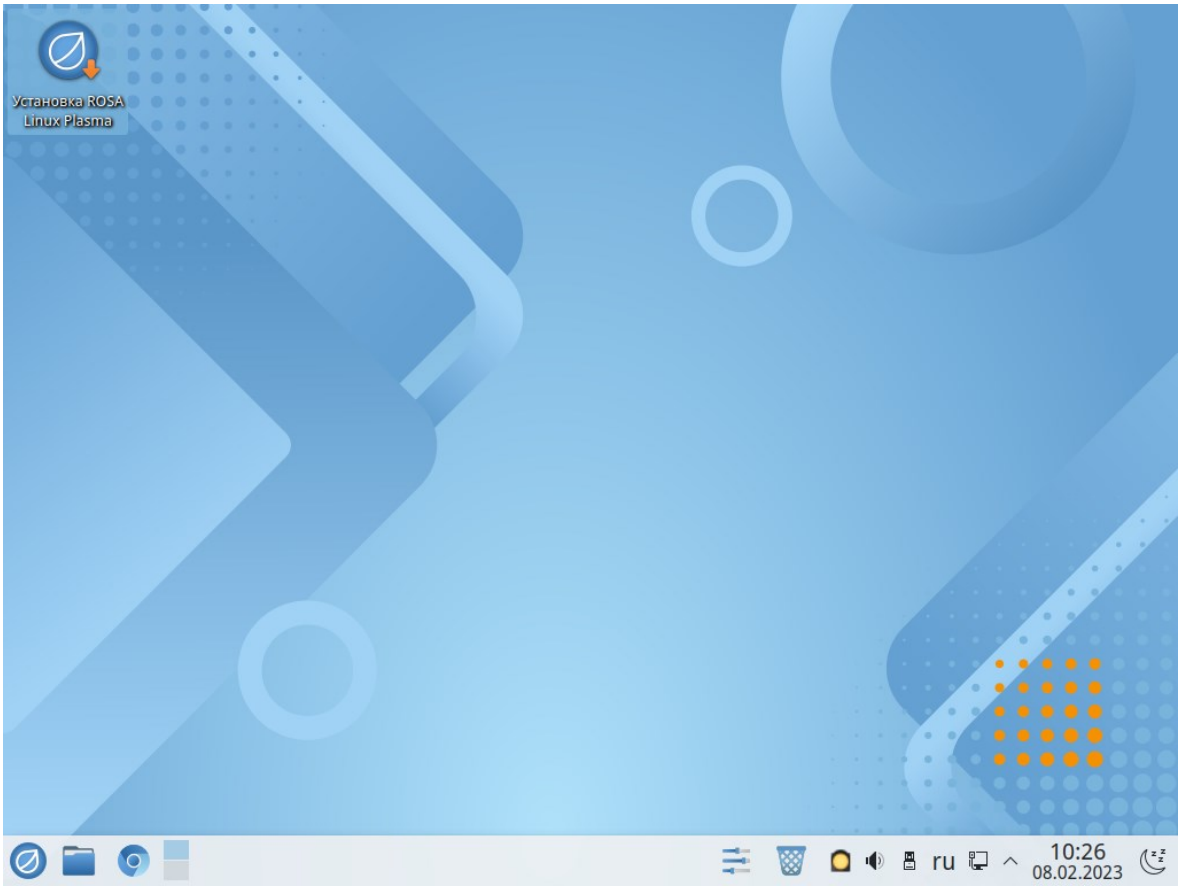


Рисунок 3 – Рабочий стол live-системы с ярлыком инсталлятора

Экран успешной проверки ОЗУ приведен ниже (рисунок 4). Для выхода из режима проверки необходимо нажать клавишу Esc, при этом произойдет перезагрузка ПК.

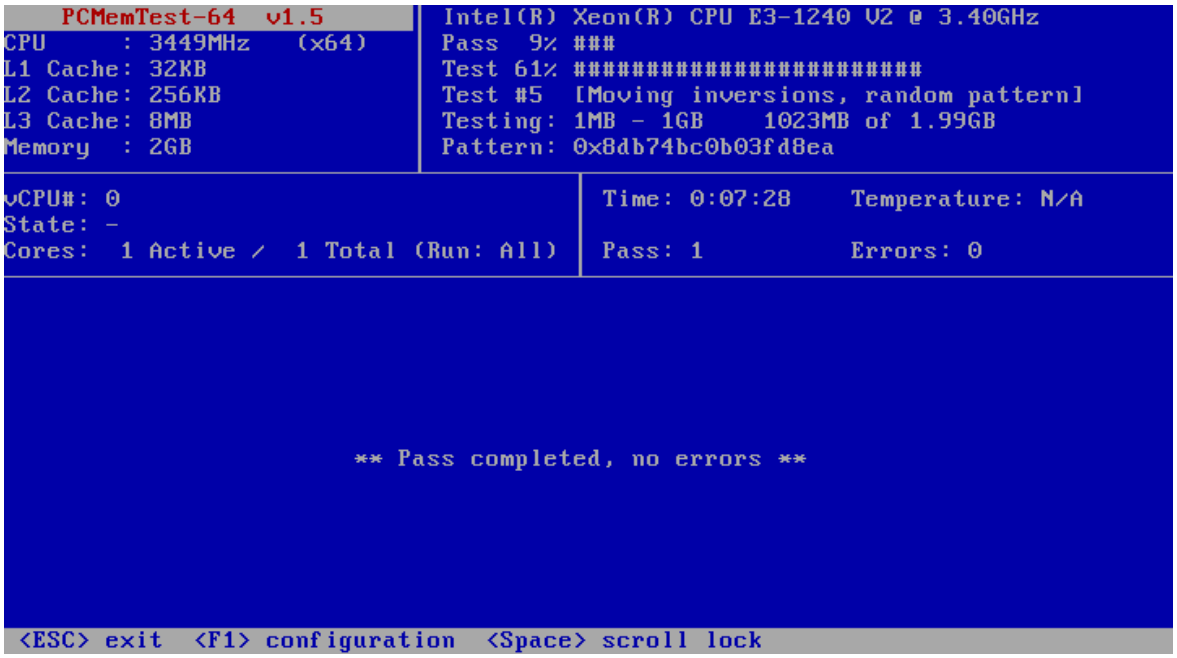


Рисунок 4 – Экран успешной проверки ОЗУ

е) Главный экран установщика ОС содержит семь разделов параметров, часть из которых заполнена автоматически, другая же часть (отмеченная красным цветом) требует вмешательства пользователя-установщика Системы (рисунок 5). Для неотмеченных красным разделов возможно тоже может понадобиться дополнительная настройка, например, при подключении ОС к локальной сети рекомендуется изменить значение "Имя узла" на имя узла, соответствующее правилам вашей сети. В случае отсутствия подключения к сети или при ошибке автоопределения необходимо также правильно установить часовой пояс.

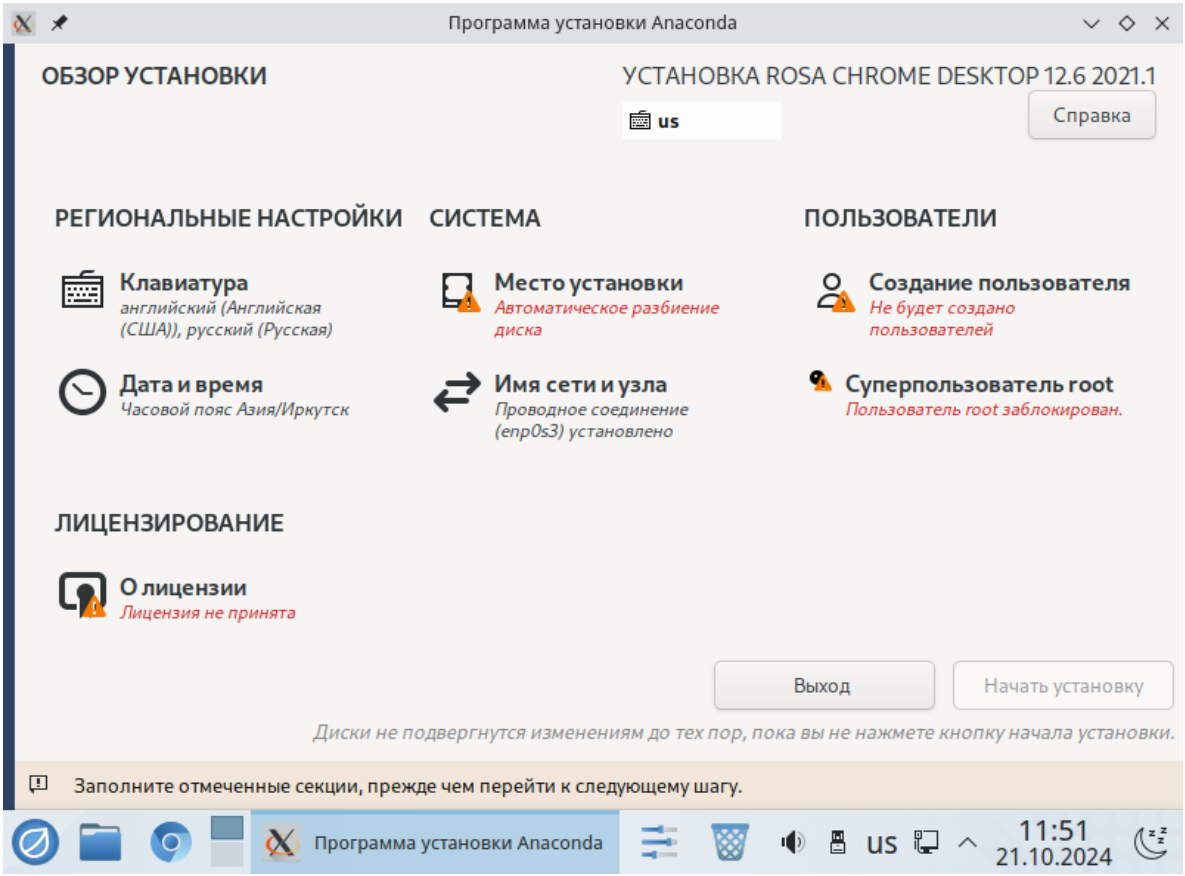


Рисунок 5 – Главный экран установщика Системы

ж) Для разметки установочного накопителя используется раздел "Место установки" (рисунок 6).

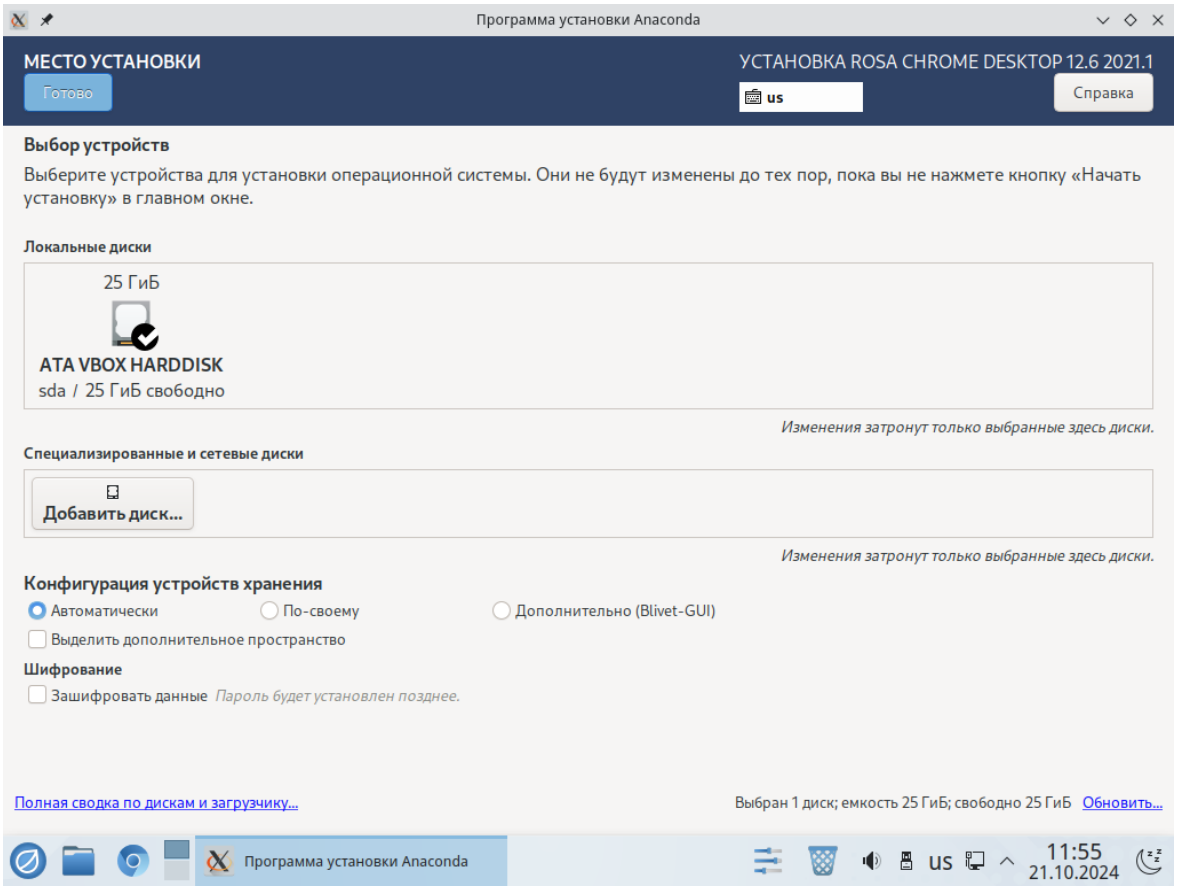


Рисунок 6 – Место установки системы

Если предполагается установка Системы на пустой носитель или же на выделенное на носителе пустое место, рекомендуется использовать автоматическую разметку. Для этого достаточно нажать на кнопку Готово, при этом, кроме технического раздела /boot/efi, будет создан отдельный корневой раздел под ОС и раздел под файлы пользователей /home, а также раздел swap для расширения оперативной памяти. Стандартная разметка дает возможность быстрого восстановления работоспособности Системы при любых её отказах, для этого достаточно переустановить Систему в корневом разделе, не форматируя раздел с пользовательскими файлами.

По умолчанию ОС устанавливается на журналируемую ФС Ext4, но в инсталляторе также имеется возможность установки на Ext3, XFS и BTRFS. Кроме того, возможно форматировать и подключать диски с NTFS.

з) Для повышения защищенности информации есть возможность установить Систему на зашифрованный диск. Подробнее об установке системы с использованием средств маскирования рассмотрено в разделе 2.3.

и) Далее ввести учетные данные первого пользователя. По умолчанию он является администратором Системы, от его имени можно вводить консольные root-команды, используя sudo, и изменять критичные настройки Системы (рисунок 7). Подробнее о пользователях и администраторах см. раздел 3.2.

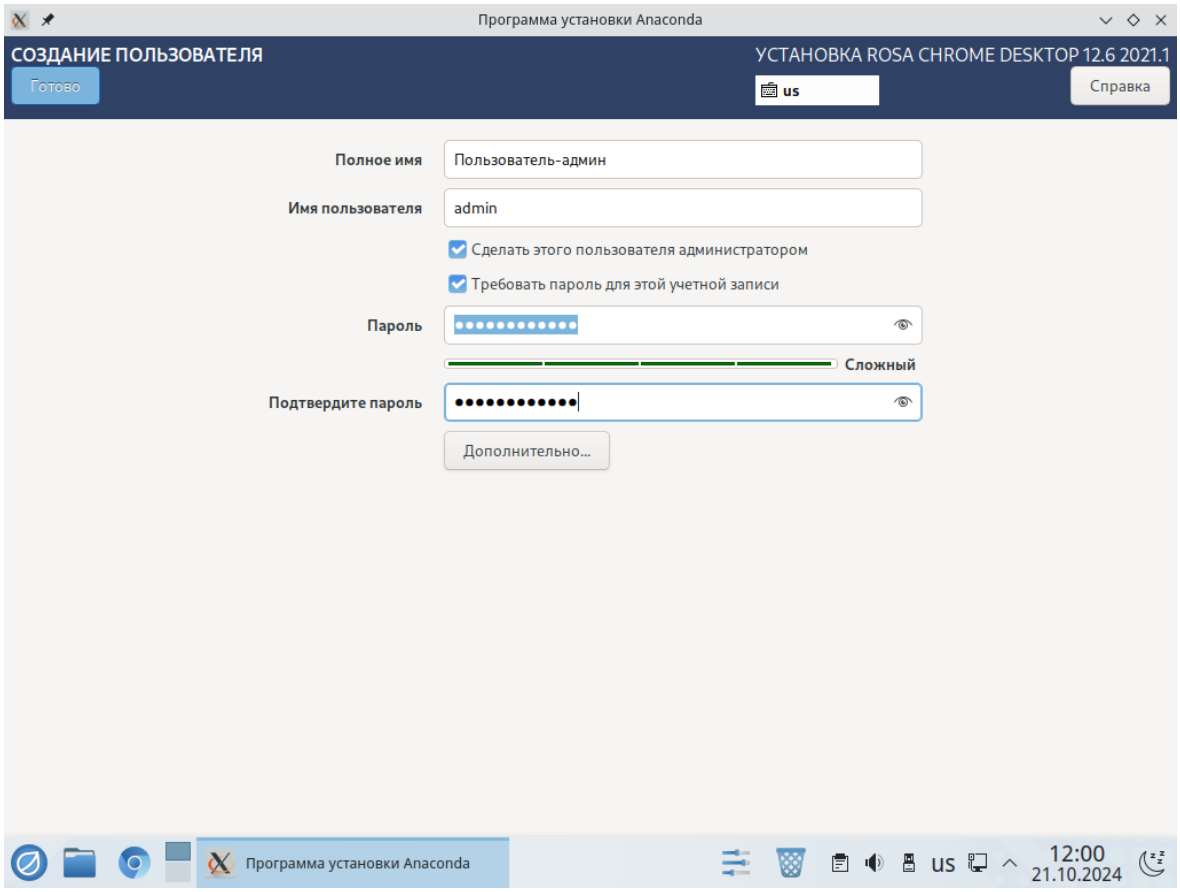


Рисунок 7 – Создание учетной записи пользователя

При вводе пароля контролируется его сложность: по длине, разнообразию знаков и по соответствию спискам часто применяемых паролей, но при этом сохраняется возможность использования небезопасных паролей при двойном нажатии на кнопку Готово.

к) Если при создании пользователя снять флажок "Сделать этого пользователя администратором", то обязательным останется следующее поле "Суперпользователь root", в которое необходимо ввести пароль суперадминистратора. Такая возможность оставлена для совместимости, при этом вводимый по умолчанию пользователь-администратор с правом sudo считается более безопасным вариантом.

л) После того как все обязательные пункты для установки заполнены, можно начать собственно установку ОС на ПК нажатием на кнопку Начать установку на главном экране установщика (рисунок 5).

м) Установка ОС может занять некоторое время, по завершении которого отобразится соответствующее сообщение (рисунок 8). После чего необходимо выполнить перезагрузку ПК, нажав кнопку Перезагрузка системы, в процессе которой необходимо извлечь установочный компакт-диск или флеш-накопитель.

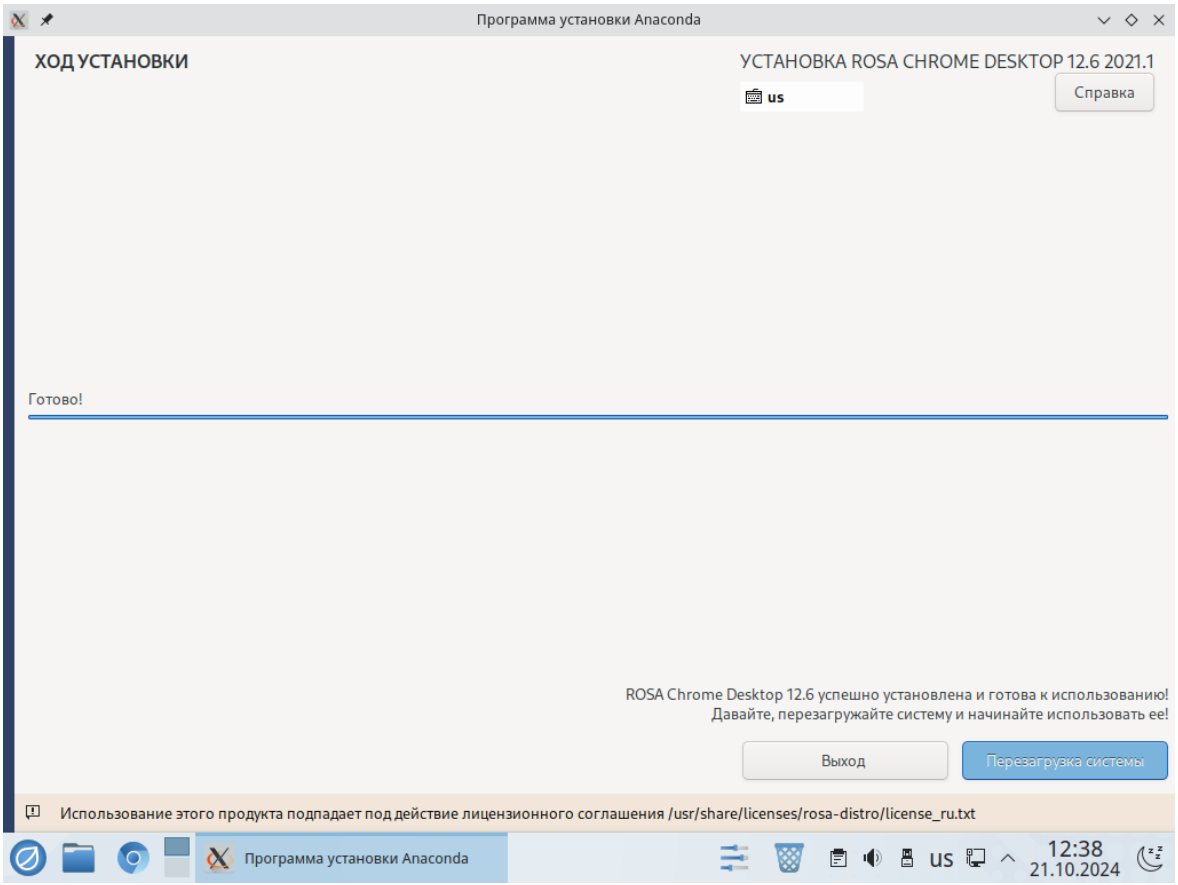


Рисунок 8 – Завершение процесса установки системы

н) После перезагрузки необходимо ввести (выбрать) логин и пароль учетной записи для выполнения входа в Систему. Если ОС была установлена на зашифрованные разделы, также понадобится ввести пароль для их расшифровки.

2.2.3 Установка серверной версии ОС

Установка серверной версии ОС осуществляется полностью аналогично установке на рабочую станцию (п. 2.2.2), за исключением того, что у серверной версии отсутствует live-режим. В меню начальной загрузки следует выбрать пункт "Установка ROSA Linux" для установки без предварительной загрузки ОС.

2.2.4 Парольная защита загрузчика ОС

Для повышения уровня безопасности возможно ограничить доступ к редактированию параметров загрузки, установив пароль на загрузчик GRUB2.

Этапы настройки пароля:

а) Генерация хеша пароля командой:

```
sudo grub2-mkpasswd-pbkdf2
```

Будет предложено ввести и подтвердить пароль. В результате отобразится хеш пароля, например:

```
Хэш PBKDF2 вашего пароля:
grub.pbkdf2.sha512.10000.7730DD2A5B6B5C31CD447BB9A8C313DC...
```

б) Создание конфигурационного файла с ограниченными правами доступа:

```
(umask 077 && touch /boot/grub2/user.cfg )
```

в) Запись сгенерированного хеша пароля в файл /boot/grub2/user.cfg:

```
echo
"GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.7730DD2A5B6B5C31CD447BB
9..." > /boot/grub2/user.cfg
```

г) Обновление конфигурации GRUB2 применением изменений:

```
sudo update-grub2
```

После выполнения данных действий при попытке редактирования записей загрузчика GRUB2 (например, при нажатии клавиши E в меню загрузки) потребуется ввод пароля. Имя пользователя — root, пароль соответствует ранее заданному при генерации хеша.

2.3 Установка с использованием средств маскирования информации

2.3.1 Общие вопросы

В этом разделе будут рассмотрены только особенности установки ОС на компьютер пользователя с использованием средств маскирования информации на носителях. В документации к защищенным ОС используется термин "шифрование", если это шифрование является сертифицированным для защиты информации. В данном документе для несертифицированных к применению в РФ алгоритмов и программ шифрования используется термин "маскирование информации". Такое маскирование не обеспечивает гарантированной защиты информации, но значительно затрудняет неавторизованный доступ к ней.

2.3.2 Разметка носителя с маскированием информации

Для повышения защищенности информации есть возможность установить Систему с использованием маскирования информации, при этом разметку носителя можно осуществить автоматически, включив флажок "Зашифровать данные" или вручную.

Для режима разметки вручную в окне "Место установки" нужно выбрать в разделе "Конфигурация устройств хранения" переключатель "По-своему" и нажать на кнопку Готово, а потом разметить носитель с отдельным незашифрованным /boot (и /boot/efi для efi-BIOS). Корневой раздел Системы, swap-раздел и пользовательский раздел /home нужно при этом зашифровать (замаскировать), установив соответствующий флажок в поле "Зашифровать данные", затем кнопки Применить и Готово (рисунок 9).

Следует обратить внимание, что замаскированные разделы обозначены как LUKS-разделы. Рекомендуется использовать систему маскирования luks2.

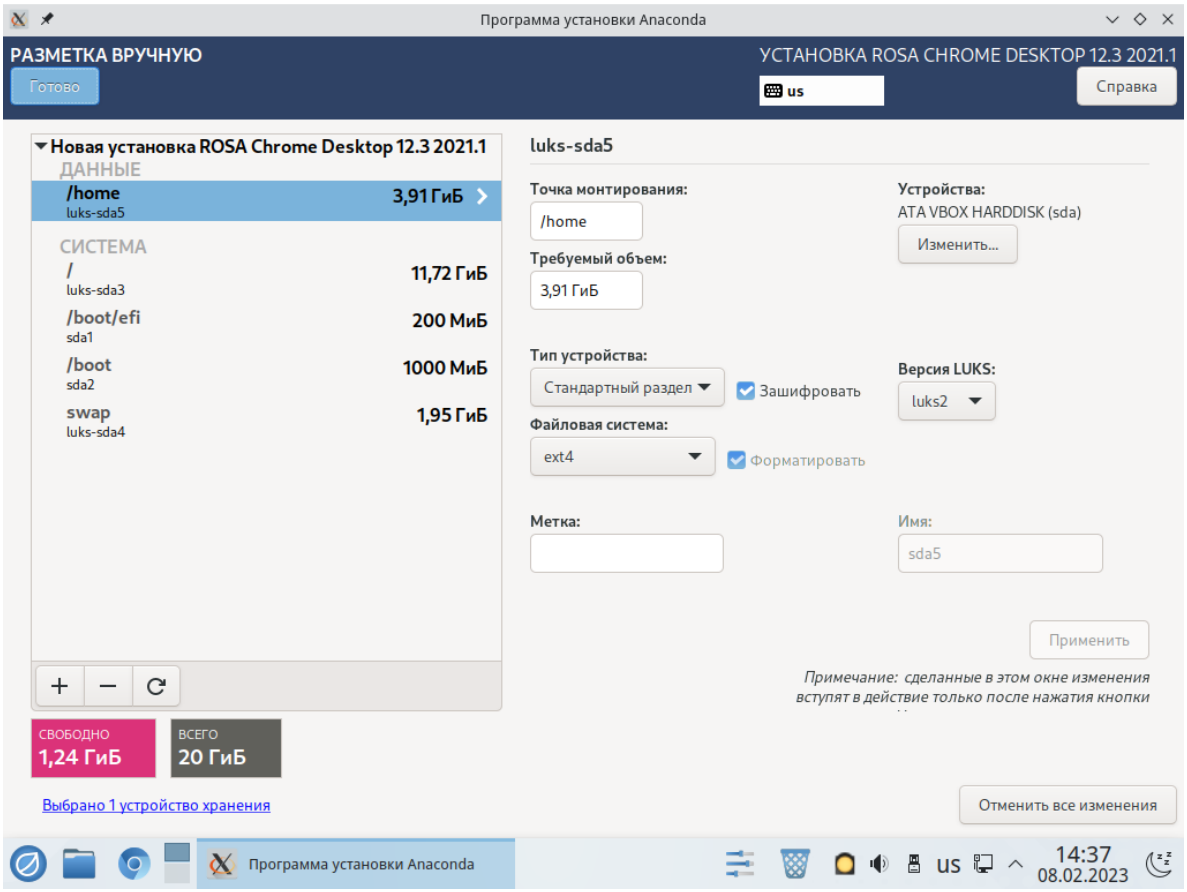


Рисунок 9 – Ручная разметка диска для использования маскирования

После этого можно выполнить установку в обычном режиме. При каждой загрузке Системы, использующей замаскированные разделы, у пользователя запрашивается пароль для расшифровки. Без ввода правильного пароля доступ к информации на носителях будет блокирован, причем эта блокировка сохраняется и при переносе носителя информации на другой ПК.

2.3.3 Поддержка tpm2 для автоматического ввода паролей

Современные СВТ с архитектурой x86_64 примерно с 2016-го года оснащаются встроенным модулем хранения конфиденциальной информации tpm2, который можно использовать для разблокировки замаскированного раздела LUKS. Пароль расшифровки маскированных разделов при этом будет храниться в tpm2 и вводиться автоматически. Пользовательская информация на изъятом из компьютера с tpm2 носителе будет недоступна при монтировании этого носителя на другом компьютере.

Для использования tpm2 в Системе, использующей разделы с маскированием информации, нужно ввести команду и следовать дальнейшим инструкциям:

```
sudo luksunlock
```

Для возврата к ручному вводу паролей используется команда:

```
sudo lukslock
```

2.4 Установка на виртуальную машину

Образ ОС подготовлен для установки и работы на виртуальных машинах (далее – ВМ) в системах виртуализации ROSA Virtualization, VirtualBox, QEMU/KVM и ESXi VMWARE. В ОС предустановлены пакеты для поддержки совместимости (virtualbox-guest-addidtions, spice-vdagent и open-vm-tools), лишние из которых удаляются при установке в несовместимой среде.

Установка под управлением гипервизора должна осуществляться согласно документации к этому гипервизору с учетом минимальных требований ОС к аппаратному обеспечению (п.1.4).

2.5 Автоматизированное развертывание

2.5.1 Введение в kickstart-сценарии

Автоматизированная установка позволяет выполнить полностью автоматическую установку в соответствии с преднастройками, может применяться для автоматического развертывания множества компьютеров одинаковой конфигурации.

Для автоматизированной установки создается файл с настройками – kickstart-сценарий – сценарий быстрого развертывания. Затем в параметры cmdline ядра записывается путь к этому файлу, по DHCP автоматически настраивается сеть, а модуль anaconda-dracut в initramfs (образ файловой

системы, загружаемый в оперативную память вместе с ядром) загружает этот kickstart-сценарий и сохраняет в файле /run/install/ks.cfg.

При установке в любом из режимов, в том числе в интерактивном графическом, в установленной Системе по адресу /root/anaconda-ks.cfg сохраняется kickstart-сценарий, описывающий произведенную установку, который можно взять за основу для пользовательского сценария в дальнейшем.

2.5.2 Типовой сценарий автоматизированной установки

Для реализации типового сценария автоматизированной установки нужно создать файл, например, с именем a.ks, со следующим содержимым:

```
cmdline
# https://bugzilla.redhat.com/show_bug.cgi?id=1874434
liveimg --url=file:///dev/mapper/live-base

# Автоматически принять лицензионное соглашение
eula --agreed

# System language
lang ru_RU.UTF-8

# Keyboard layouts
keyboard --xlayouts=us,ru --switch=grp:alt_shift_toggle

# Network information
network --hostname=localhost.localdomain

# SELinux configuration
#selinux --disabled

# System services
services --enabled="chronyd"

autopart
# Partition clearing information
clearpart --all --initlabel

# System timezone
timezone Europe/Moscow --utc
```

```
    # Passwords
    rootpw 123456
    user --groups=wheel,users --name=rosa --password=123456

    %anaconda
    pwpolicy root --minlen=6 --minquality=1 --notstrict --
nochanges --notempty
    pwpolicy user --minlen=6 --minquality=1 --notstrict --
nochanges --emptyok
    pwpolicy luks --minlen=6 --minquality=1 --notstrict --
nochanges --notempty
    %end

    reboot --eject
```

Примечание – С документацией по kickstart-сценариям можно ознакомиться по адресу https://pykickstart.readthedocs.io/en/latest/kickstart-docs.html.

Код приведенного выше сценария состоит из следующих командных строк:

а) Выполняется полностью автоматизированная установка: запускается X-сервер, автоматически запускается установка, не требующая настройки (возможны варианты без X-сервера и установка по VNC):

```
    cmdline
```

Копируется на накопитель Система из LiveCD (также возможна установка из сетевого репозитория):

```
    liveimg --url=file:///dev/mapper/live-base
```

Устанавливается язык ОС ru_RU.UTF-8 (в файле /etc/locale.conf):

```
    lang ru_RU.UTF-8
```

Добавляются в ОС две раскладки клавиатуры: us и ru, а переключение раскладок настраивается на комбинацию клавиш `Alt+Shift`:

```
    keyboard --xlayouts=us,ru --switch=grp:alt_shift_toggle
```

Включается служба systemd chronyd (можно перечислить несколько служб через запятые без пробелов):

```
    services --enabled="chronyd"
```

Пример отключения sshd:

```
    services --disabled="sshd"
```

Выполняется полностью автоматизированная разметка диска с удалением всех данных (возможно прописать детализированные инструкции по разметке диска):

```
clearpart --all --initlabel
```

Устанавливается часовой пояс "Европа/Москва", а аппаратные часы (BIOS) будут выставлены по Гринвичу (рекомендуется):

```
timezone Europe/Moscow --utc
```

Устанавливается последовательность символов, например, "123456", в качестве root-пароля:

```
rootpw 123456
```

Создается пользователь с именем, например, "rosa", добавляется в группы wheel и users; устанавливается пароль пользователя, например, "123456":

```
user --groups=wheel,users --name=rosa --password=123456
```

Примечание – Группа wheel – специальная группа пользователей, используемая в некоторых системах UNIX для контроля доступа к команде su, которая позволяет пользователю маскироваться под другого пользователя (обычно суперпользователя).

Пароли можно записать в зашифрованном виде, например, установить пароль пользователю root:

```
rootpw --iscrypted
$6$BHWkSXIw6TgPrdvJ$V1ARg/yAP.kRe5mo5FawOOvf7SRm/GDM.Ly6RGGmSBR3
Zc81/XzYlpoDnYKFMMPZodKBIVyBvjJ9evM1sey0E/
```

Хеш, приведенный выше, является результатом выполнения команды:

```
python3 -c "import crypt; print(crypt.crypt(input('clear-text
pw: '), crypt.mksalt(crypt.METHOD_SHA512)))"
```

Данная команда позволяет получить хеш введенного с клавиатуры пароля. Таким образом, можно избежать хранения пароля в открытом виде. Например, хеш пароля "123456" выглядит следующим образом:

```
user --groups=wheel,users --name=rosa --
password=$6$BHWkSXIw6TgPrdvJ$V1ARg/yAP.kRe5mo5FawOOvf7SRm/GDM.Ly
6RGGmSBR3Zc81/XzYlpoDnYKFMMPZodKBIVyBvjJ9evM1sey0E/ --iscrypted
```

к) По завершении установки необходимо перезагрузить ПК (в процессе перезагрузки необходим установочный носитель):

```
reboot --eject
```

Примечание – При установке на ВМ рекомендуется использовать установку с ISO-образа, подключенного как DVD, т.к. гипервизор умеет отключать виртуальный

привод по команде извлечения диска из гостевой Системы. В таком случае после перезагрузки Система загрузится не с образа, а с диска с установленной ОС.

Параметры в cmline ядра и в kickstart-сценарии дополняют друг друга и могут использоваться в различных комбинациях.

2.5.3 Дополнительные действия в kickstart-сценариях

В kickstart-сценарий могут быть добавлены произвольные действия, выполняемые в интерпретаторе shell после завершения основной установки. Это позволяет настраивать Систему сразу после инсталляции.

Для выполнения действий в установленной Системе (в chroot) используется секция %post:

```
%post
<команды>
%end
```

Пример: запись даты и времени установки в файл /var/log/installtime.

```
%post
LC_ALL=C date > /var/log/installtime
%end
```

Если необходимо выполнить действия вне установленной Системы (в окружении LiveCD), в секцию %post добавляется параметр --nochroot:

```
%post –nochroot
LC_ALL=C date > "${INSTALL_ROOT}/var/log/installtime"
%end
```

Примечание – Дополнительно возможно загрузить внешние файлы с помощью утилит curl, wget и других средств сетевого доступа, при наличии доступа к локальному или внешнему веб-серверу.

Удаление и установка пакетов могут быть выполнены также внутри секции %post:

– Удаление ненужных пакетов:

```
%post
sudo dnf -q -y remove firefox-esr52
%end
```

– Установка дополнительных пакетов:

```
%post
dnf -q -y install sconfigs-detailed-audit
```

```
%end
```

Прямая установка пакетов через секцию %packages в данной конфигурации не поддерживается. Рекомендуется использовать утилиту dnf с ключом "-q" для подавления лишнего вывода, что повышает стабильность автоматизированной установки.

Для тестирования сценария автоматизированной установки можно воспользоваться встроенным HTTP-сервером на машине оператора:

а) Определить IP-адрес оператора:

```
ip a
```

б) Перейти в директорию с файлом сценария (например, a.ks) и запустить сервер:

```
python3 -m http.server
```

Примечание – В качестве альтернативы возможно использование других веб-серверов, например, nginx или Apache httpd.

в) На целевой машине при загрузке LiveCD в командную строку ядра необходимо добавить:

```
systemd.unit=anaconda.target
inst.ks=http://192.168.1.173:8000/a.ks
```

где:

– 192.168.1.173 — IP-адрес машины оператора,
– 8000 — порт запущенного HTTP-сервера.

Также рекомендуется добавить параметр inst.text.

Данный параметр позволит запустить консольный режим установщика вместо графического. Установка будет произведена автоматически без участия оператора.

2.6 Сетевая загрузка с возможностью установки

2.6.1 Технология Intel PXE

Возможность загрузки ОС по сети реализована с помощью стандартной технологии PXE, которая состоит из следующих этапов:

а) Встроенный в BIOS компьютера загрузчик PXE:

– получает IP-адрес по протоколу BOOTP (протокол, родственный DHCP);

– скачивает по протоколу TFTP с сервера TFTP файлы загрузчика, в т.ч. сам загрузчик и конфигурационный файл с описанием меню загрузки, а также элементы vmlinuz и initrd в качестве минимальной версии ОС;

– запускает минимальную версию ОС с указанными в конфигурационном файле параметрами.

б) Минимальная версия ОС, запущенная на клиентском компьютере:

– скачивает указанные в параметрах запуска initrd установочный образ и дополнительные компоненты полной версии ОС по протоколу HTTP с установленного на сервере PXE веб-сервера, сохраняя их в оперативной памяти;

– запускает скачанную полную версию ОС на клиенте.

Таким образом, на клиентском компьютере запускается тот же установочный образ LiveCD, что и при загрузке с USB-накопителя или DVD.

Следует учесть, что для сетевой загрузки ОС на клиентской системе требуется на 2 Гб оперативной памяти больше, чем для обычной установки.

2.6.2 Начальная настройка сервера для работы PXE

Для сетевой загрузки и установки на рабочих станциях необходим сервер, с которого будут загружаться образы ОС, и локальная сеть, по которой будет происходить загрузка. В качестве сервера можно использовать как установленную локально или в виртуальной машине версию ОС для рабочей станции, так и серверную версию ОС.

Для того чтобы ОС могла работать в качестве сервера PXE, необходимо установить и настроить серверные пакеты. Для настройки понадобятся сервера DHCP и TFTP (простейший веб-сервер для загрузки по HTTP уже предустановлен), а также загрузчик syslinux:

```
dnf install dhcp-server tftp-server syslinux
```

2.6.3 Настройка сервера DHCP (BOOTP)

Конкретная настройка DHCP-сервера специфична для каждой локальной сети. В качестве примера можно рассмотреть по шагам простейший случай:

а) установить на сервере PXE для текущего сетевого соединения статический IP-адрес 192.168.1.1 и маску подсети 255.255.255.0;

б) заменить содержимое файла настройки /etc/dhcpd.conf на следующее:

```
option domain-name "rosa.lan";
option domain-name-servers ns1.rosa.lan, ns2.rosa.lan;
default-lease-time 3600;
```

```
     max-lease-time 7200;
     authoritative;
allow booting;
     allow bootp;
     filename "pxelinux.0";
     subnet 192.168.1.0 netmask 255.255.255.0 {
      option routers 192.168.1.1;
      option subnet-mask 255.255.255.0;
      option domain-search "rosa.lan";
      option domain-name-servers 192.168.1.1;
      range 192.168.1.10 192.168.1.100;
     }
```

в) включить сервис DHCP командой:

```
     sudo systemctl enable --now dhcpd
```

г) статус запущенного сервера (должен быть active) можно проверить командой:

```
     systemctl status dhcpd
```

Правильность подключения можно проверить, войдя в сеть с клиентского компьютера: он должен получить от сервера DHCP IP-адрес 192.168.1.10.

2.6.4 Настройка веб-сервера

В предустановленном на образах ОС языке Python есть простой веб-сервер, который можно использовать для тестовой PXE-установки (потребуется ISO-образ ОС), последовательно выполнив следующие команды:

а) создать папку для веб-сервера:

```
     mkdir -p "$HOME/webserver"
```

б) перейти в папку $HOME/webserver:

```
     cd "$HOME/webserver"
```

в) создать папку для ISO-образа:

```
     mkdir -p ISO
```

г) монтировать папку для ISO-образа:

```
     sudo mount -o loop "<путь к ISO-образу>"
"$HOME/webserver/ISO"
```

д) запустить HTTP-сервер из папки:

```
python3 -m http.server
```

Далее необходимо скопировать в папку $HOME/webserver файл kickstart-сценария a.ks, создание которого описано в п.2.5.2.

Консоль, в которой будет работать веб-сервер, следует оставить открытой. Проверить правильность его настройки можно, открыв на компьютере браузер по адресу http://192.168.1.1:8000.

В браузере должны быть выведены файлы из директории веб-сервера, в том числе должны быть доступны папка ISO, ее содержимое и файл kickstart-сценария a.ks, а в консоли веб-сервера должен быть выведен лог запросов к нему.

2.6.5 Настройка сервера TFTP

Для настройки сервера TFTP нужно выполнить следующие действия:

а) скопировать в папку сервера TFTP компоненты первоначального загрузчика:

```
( cd /usr/share/syslinux && cp -v menu.c32 pxelinux.0 ldlinux.c32 libutil.c32 /var/lib/tftpboot/ )
```

б) скопировать минимальную версию ОС из ISO-образа в папку TFTP-сервера (от имени пользователя, но не root):

```
( cd "$HOME/webserver/ISO/isolinux" ; sudo cp -v initrd0.img vmlinuz0 /var/lib/tftpboot/ )
```

в) создать папку для дополнительных конфигурационных файлов TFTP-сервера:

```
mkdir -p /var/lib/tftpboot/pxelinux.cfg
```

г) открыть на редактирование файл /var/lib/tftpboot/pxelinux.cfg/default (будет создан, если не существует):

```
nano /var/lib/tftpboot/pxelinux.cfg/default
```

Содержимое файла /var/lib/tftpboot/pxelinux.cfg/default:

```
default menu.c32
prompt 0
timeout 300
ONTIME local
menu title #================ PXE BOOT ===============#

 label 1
```

```
      menu label ^1) Automatically install ROSA
      kernel /vmlinuz0
      append   initrd=/initrd0.img   systemd.unit=anaconda.target
enforcing=0
root=live:http://192.168.1.1:8000/ISO/LiveOS/squashfs.img
inst.text inst.ks=http://192.168.1.1:8000/a.ks

      label 2
      menu label ^2) Install ROSA
      kernel /vmlinuz0
      append           initrd=/initrd0.img           enforcing=0
root=live:http://192.168.1.1:8000/ISO/LiveOS/squashfs.img
```

Для сохранения данных и выхода из редактора nano нажимают клавиши `Ctrl+O` и `Ctrl+X` соответственно.

В строках, начинающихся с append, определены параметры cmdline ядра для вариантов загрузки в меню. В первом случае запускается автоматизированная консольная установка по kickstart-сценарию, а во втором случае запускается LiveCD. При отсутствии выбора со стороны пользователя какого-либо пункта меню в течение 300 секунд будет произведена загрузка первого пункта.

Следует обратить внимание, что приведенный выше kickstart-сценарий автоматически стирает весь диск, т.е. в приведенной конфигурации, если не выбрать второй пункт меню, будет произведена автоматическая установка с форматированием диска. При запуске ОС по PXE можно выбрать нужный пункт меню. При необходимости можно адаптировать меню под свои потребности (дополнительная информация доступна по ссылке https://wiki.syslinux.org/wiki/index.php?title=PXELINUX).

д) запустить и добавить в автозагрузку TFTP-сервер:

```
      sudo systemctl enable --now tftp
```

е) выполнить настройки BIOS клиентской машины на загрузку по сети (PXE) и произвести загрузку.

2.6.6 Настройка сети в LiveCD

По умолчанию при запуске LiveCD, включая режим работы исключительно установщика ОС, настройка сети осуществляется автоматически по протоколу DHCP.

В случае необходимости задания статических сетевых параметров соответствующая конфигурация может быть передана в ядро через параметр cmdline, используя синтаксис, поддерживаемый модулем dracut network-legacy.

Пример задания параметров сети вручную:

```
ip=192.168.122.50::192.168.122.1:255.255.255.0:::off
```

где:

– 192.168.122.50 — IP-адрес сетевого интерфейса;
– 192.168.122.1 — шлюз по умолчанию;
– 255.255.255.0 — маска подсети;
– off — отключение автоматической настройки через DHCP.

Примечание – Дополнительные сведения о параметрах и форматах можно найти на справочной странице dracut.cmdline(7).

2.7 Графическая установка с использованием VNC

Графическая установка с применением технологии VNC (Virtual Network Computing) позволяет запускать интерфейс установщика ОС на удалённой машине оператора. Этот способ особенно полезен в случаях, когда установка производится на удалённый компьютер, не имеющий локальных устройств ввода и отображения (монитор, клавиатура, мышь), либо физический доступ к нему затруднён.

Примечание – При использовании установки через VNC запускается встроенный VNC-сервер, который может быть доступен без фильтрации трафика и, в зависимости от конфигурации, без пароля. Не рекомендуется запускать установку по VNC в незащищённых или общедоступных сетях без дополнительных мер безопасности.

Необходимые условия для выполнения установки через VNC:

– целевая машина должна иметь установленный VNC-клиент;
– целевая машина, на которую производится установка ОС, должна быть доступна по сети и поддерживать загрузку с LiveCD или по PXE;
– сеть должна быть настроена по DHCP или вручную, с обеспечением двусторонней сетевой доступности между клиентом и целевым компьютером.

Установка ОС через VNC выполняется в несколько этапов:

а) Установка клиента TigerVNC на целевом ПК командой:

```
sudo dnf install /usr/bin/vncviewer
```

б) Запуск клиента TigerVNC в режиме ожидания входящего подключения на целевом ПК:

```
vncviewer -listen 4444
```

где 4444 — произвольно выбранный свободный порт, который будет использоваться для подключения.

в) Получение IP-адреса целевого компьютера, на котором выполняется vncviewer, командой:

```
/sbin/ip a
```

г) При загрузке целевого компьютера с LiveCD или по сети (PXE-загрузка) необходимо изменить параметры командной строки ядра. В режиме ввода параметров загрузки (boot prompt) добавить следующее:

```
systemd.unit=anaconda.target                    inst.vnc\
inst.vncconnect=192.168.1.173:4444
```

где:

– 192.168.1.173 — IP-адрес целевого компьютера;

– 4444 — порт, указанный при запуске vncviewer.

д) Завершение установки и запуск графического интерфейса. После загрузки и автоматической настройки сети установщик подключится к оператору по указанному адресу. На целевом компьютере автоматически откроется окно графического установщика ОС РОСА "ХРОМ", идентичное локальному интерфейсу установки. Можно приступить к стандартному графическому процессу установки (об этом подробнее см. раздел 2.2 Локальная установка), управляя им удалённо.

2.8 OEM-установка

OEM-установка предназначена для массового развёртывания ОС на однотипные компьютеры. В этом методе на жёсткий диск целевой машины побайтово копируется ISO-образ, а не клон уже установленной Системы. После копирования машины могут быть переданы конечным пользователям без дополнительной настройки.

При первом запуске администратор подтверждает установку ОС, после чего у него отсутствует возможность вмешаться в процесс. После завершения установки будет предложено создать пользователей и подтвердить лицензионное соглашение.

Преимущества OEM-установки:

– размер образа для OEM-установки равен размеру стандартного ISO, что значительно меньше по сравнению с клон-образом уже установленного диска;

– после подтверждения пользователем установки исходный ISO-образ остаётся на первом разделе, что позволяет при неповреждённой файловой системе вернуть Систему к состоянию "первого старта";

– создание ISO-образа для OEM-установки происходит быстрее и проще, что снижает вероятность ошибок.

OEM-образ формируется в два этапа:

а) Сборка образа на ABF. Необходимо, чтобы в сборке присутствовал пакет oem-install. С 2025 года данный пакет включается во все стандартные образы ОС РОСА "ХРОМ". Он не мешает установке через Anaconda и будет автоматически удалён на этапе initial-setup.

б) Пересборка с помощью скрипта oem-iso-rebuilder. Основная задача этого скрипта — заменить конфигурационный файл загрузчика GRUB2 в ISO-образе, чтобы при загрузке запускался oem-install вместо Live OS или Anaconda. Кроме того, скрипт меняет таблицу разделов на GPT и даёт возможность добавить файлы, влияющие на поведение OEM-установки.

Пример пересборки:

```
    sudo oem-iso-rebuilder\
./ROSA_2021.1_PLASMA5_x86_64_53042.iso
```

После запуска утилита предложит приостановить выполнение, чтобы добавить собственные конфигурационные файлы. По завершении будет получен новый образ с расширением .oem.iso.

Установка OEM-образа производится побайтовым копированием ISO-образа на носитель, для этого можно использовать утилиту dd:

```
    dd if=./ROSA_2021.1_PLASMA5_x86_64_53042.oem.iso of=/dev/sda
bs=4M
```

При пересборке ISO допускается добавить в его корень следующие файлы:

– oem-install.cfg — конфигурация разметки дисков, например:

```
    PARTITONS=btrfs
    ROOTSIZE=70000
    SWAPSIZE=10000
    EFISIZE=200
```

Подробнее обо всех параметрах можно узнать, выполнив команду:

```
    /usr/libexec/oem-install --help
```

– anaconda-ks.cfg — kickstart-файл, который обычно создаётся Anaconda, будет скопирован в /root установленной Системы;

– oem-install.css — стили интерфейса GUI для oem-install, шаблон доступен в /usr/share/oem-install/oem-install.css.

2.9 Установщик iso2img

Утилита iso2img представляет собой простой скриптовый инструмент, позволяющий в один шаг установить ОС РОСА "ХРОМ" из ISO-образа в образ виртуальной машины (qcow2, vmdk, vdi и др.). При этом будет создан пользователь rosa с паролем rosa и правами администратора.

Установка и использование утилиты iso2img:

```
sudo dnf install iso2img
sudo iso2img Rosa.iso
```

В результате будет получен, например, файл Rosa.qcow2, который можно запустить напрямую:

```
qemoo Rosa.qcow2
```

Для получения дополнительных параметров необходимо обратиться к справке через команду:

```
iso2img –help
```

2.10 Подготовительные процедуры после установки

После завершения установки ОС необходимо выполнить ряд подготовительных процедур, направленных на обеспечение безопасной и корректной работы Системы. Перечень мероприятий зависит от ролей, которые будет выполнять данная Система, требований к защите информации и политики безопасности организации.

Первоначальная настройка включает в себя следующие административные действия:

– Настройка системы аудита. Определяется перечень критически важных событий, подлежащих регистрации. По умолчанию активировано отслеживание основных системных событий. При необходимости следует расширить список контролируемых операций и определить действия при обнаружении инцидентов безопасности. Об этом подробнее см. разделы 7.6 и 7.5.7 данного Руководства.

– Настройка поведения служб при фиксации событий безопасности. Устанавливаются параметры реакции на критические события: запись в журнал, уведомления, обработка правилами службы auditd. Об этом подробнее см. разделы 7.6.3 - 7.6.4.

– Настройка параметров хранения журналов аудита. Об этом подробнее см. раздел 7.6.3.

– Установка пароля на загрузчик GRUB2. Ограничение доступа к параметрам загрузки делает невозможным изменение загрузочных настроек

неавторизованными пользователями. Об этом подробнее рассмотрено в разделе 2.2.4.

– Создание учетных записей, групп и ролей. Формируется структура пользователей и групп в соответствии с функциональными обязанностями. Для каждой категории устанавливаются необходимые уровни доступа. Об этом подробнее см. главу 4 и раздел 3.2.7.

– Настройка прав доступа к данным. Для создаваемых каталогов и файлов следует определить корректные права и владельцев, предотвращая несанкционированный доступ. Об этом подробнее см. раздел 7.1.

– Определение политики установки и запуска программ. Настраиваются допустимые источники пакетов, методы установки и правила запуска системных и прикладных компонентов. Об этом подробнее см. разделы 3.6 и 3.4.1.

– Настройка механизмов идентификации и аутентификации. Устанавливаются требования к паролям, сроки их действия, правила блокировки учётных записей и ограничения по числу неуспешных попыток входа. Об этом подробнее см. разделы 4.2 и 4.3.

– Организация резервного копирования. Создаются резервные копии системных настроек, данных и журналов. Определяется периодичность и набор инструментов для резервирования. Об этом подробнее см. раздел 7.2.

– Настройка доверенных серверов точного времени. Корректное время необходимо для журналирования, сетевых взаимодействий и систем безопасности; рекомендуется использовать chrony. Об этом подробнее см. раздел 5.7.

– Проверка целостности системных файлов. Настраиваются механизмы контроля целостности, позволяющие выявлять изменения критически важных компонентов. Об этом подробнее см. раздел 7.3.

– Настройка отказоустойчивости (при необходимости). При эксплуатации в критически важных средах может потребоваться настройка кластеров высокой доступности. Об этом подробнее см. раздел 7.7.

– Настройка квот и ограничений ресурсов. Определяются допустимые значения использования дискового пространства и памяти пользователями или группами. Об этом подробнее в разделе 3.7.5.

– Настройка ограничений пользовательских сеансов. Устанавливаются параметры блокировки экрана, ограничения по времени бездействия и правила принудительного завершения сеансов. Об этом подробнее см. раздел 3.2.5 и раздел 7.5.8.

– Использование встроенной документации. Для получения справочной информации рекомендуется использовать встроенную систему документации `man`. Об этом подробнее см. раздел 3.3.8.

3 ОБЩИЕ СВЕДЕНИЯ О РАБОТЕ В СИСТЕМЕ

3.1 Интерфейсы Системы

Система поддерживает два основных режима работы пользовательского окружения: текстовый консольный интерфейс (виртуальные терминалы TTY) и графический интерфейс (графический сервер с рабочим столом и менеджером входа).

Серверный вариант ОС по умолчанию загружается в текстовый консольный интерфейс TTY. При необходимости допускается установка графической оболочки через менеджер пакетов dnf. Для установки рабочего стола KDE Plasma при работе с серверным вариантом следует выполнить команду:

```
sudo dnf install task-iso-plasma5
```

После установки рекомендуется включить менеджер входа, выполнив команду:

```
sudo systemctl enable --now gdm
```

Для немедленного перехода в графический режим может быть использована команда:

```
sudo systemctl isolate graphical.target
```

Для назначения графического режима в качестве режима загрузки по умолчанию следует выполнить:

```
sudo systemctl set-default graphical.target
```

Возврат к консольному режиму задаётся командой:

```
sudo systemctl set-default multi-user.target
```

ОС рабочей станции по умолчанию запускается с графическим столом KDE Plasma 5 и менеджером входа GDM. Для работы с командной строкой в графическом режиме используется эмулятор терминала. В ОС он запускается через "Меню приложений → Konsole" (подробнее см. раздел 3.1.1.1).

3.1.1 Общие сведения о работе в консоли

Взаимодействие с Системой обеспечивается специальным программным продуктом – оболочкой (shell) GNU bash.

Оболочка bash может работать как в интерактивном, так и в изолированном режимах. В первом случае программа bash взаимодействует с пользователем, во втором используется для выполнения скриптов (текстовых файлов с последовательностью команд).

При входе в консоль отображается стандартное приглашение командной строки (prompt):

```
user@hostname ~ $
```

где:

– user — имя текущего пользователя;

– hostname — имя хоста, формируемое по умолчанию случайным образом;

– @ — разделитель;

– ~ — текущий каталог (в Bash символ тильды соответствует домашнему каталогу /home/user);

– $ — символ окончания приглашения для обычного пользователя;

– # — символ окончания приглашения, указывающий на выполнение команд от имени суперпользователя root.

3.1.1.1 Эмулятор терминала

Эмулятор терминала – это программа, которая позволяет использовать терминал в графической среде ОС.

Konsole – это эмулятор терминала, предоставляемый в рамках среды рабочего стола KDE Plasma. Она является одним из основных инструментов командной строки в KDE Plasma и предоставляет мощные возможности для взаимодействия с ОС через командную строку.

Konsole также интегрирована во многие другие приложения KDE, что упрощает доступ и делает ее более удобной. Например, программы KDevelop, Kate и Dolphin используют Konsole в качестве встроенного эмулятора терминала.

Для того чтобы открыть эмулятор командной строки в графическом режиме работы ОС, следует перейти в "Меню приложений → Konsole" (рисунок 10).

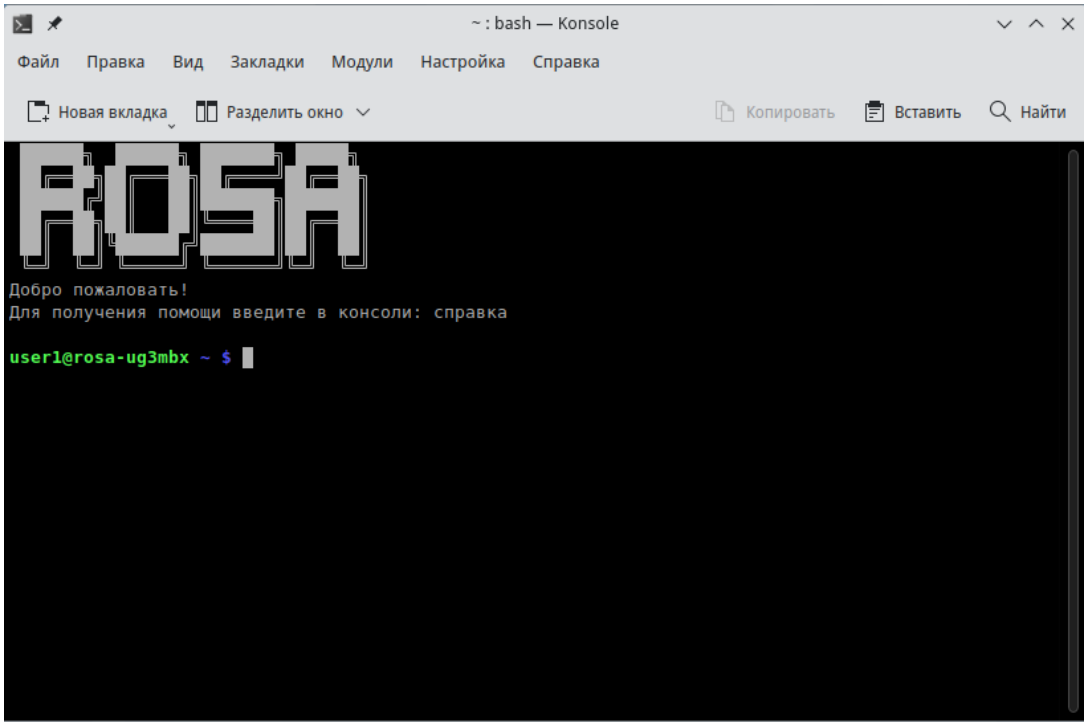


Рисунок 10 – Интерфейс окна Konsole

Интерфейс окна Konsole включает:

Меню, в котором располагаются различные команды и опции, позволяющие настроить терминал, а также дополнительные опции позволяющие взаимодействовать с другими программами.

Панель инструментов, содержащая кнопки для выполнения действий, таких как создание новой вкладки, разделение окна, копирование и вставка, а также поиск по терминалу.

Основная часть окна представляет собой область вывода, в которой отображается результат выполнения команд, текстовый вывод и другая информация.

В Системе по умолчанию доступны двенадцать виртуальных терминалов (TTY). Переключение между ними выполняется комбинациями клавиш `Ctrl+Alt+Fn`, где "n" — номер терминала от 1 до 12 (например, комбинация клавиш `Ctrl+Alt+F1` — первая консоль, а `Ctrl+Alt+F12` — двенадцатая).

3.2 Администраторы и пользователи

3.2.1 Общая информация

Как типичная Linux-система, ОС является многопользовательской. Администратор может управлять учетными записями других пользователей с помощью консольного и графического интерфейса.

В Системе существуют три типа пользователей:

Администраторы – привилегированные пользователи с полным доступом к Системе. По умолчанию на серверной ОС после ее установки всегда есть один такой пользователь root.

Локальные пользователи – непривилегированные пользователи. Их учётные записи создаёт администратор. Таким пользователям недоступны некоторые системные утилиты, они могут работать только с теми файлами и папками, к которым у них открыт доступ. Тем не менее, привилегии локального пользователя можно повысить при необходимости.

Системные пользователи – учётные записи, автоматически создаваемые Системой для работы внутренних процессов и служб. Такие пользователи нужны для повышения безопасности.

Примечание – По умолчанию при установке ОС первый, созданный в инсталляторе, пользователь является пользователем-администратором Системы.

Каждый пользователь имеет свой уникальный идентификатор пользователя UID. Идентификатор отличается в зависимости от типа пользователя:

– администратор – 0;
– обычный (локальный) пользователь – от 100;
– системный пользователь – от 1 до 100.

Чтобы упростить процесс настройки прав для новых пользователей, их объединяют в группы. Каждая группа имеет свой набор прав и ограничений. Любой пользователь, создаваемый или добавляемый в такую группу, автоматически их наследует. Если при добавлении пользователя для него не указать группу, то у него будет своя, индивидуальная группа с именем пользователя. Один пользователь может одновременно входить в несколько групп.

Информацию о каждом пользователе сервера можно посмотреть в файле /etc/passwd. Пользователи в нём перечислены в следующем формате:

```
test-user:x:1000:1000::/home/test-user:/bin/bash
```

где параметры:

– test-user – имя пользователя;

– x – признак зашифрованного пароля (хранится в /etc/shadow);

– 1000:1000 – идентификатор пользователя (UID) и идентификатор группы (GID), к которой он принадлежит;

– :: – комментарий к пользователю и дополнительная информация (например, контакты, имя и прочее);

– /home/test-user – адрес домашний папки пользователя;

– /bin/bash – указывает, что пользователь имеет доступ к командной оболочке сервера.

Для отображения только учетных записей пользователей можно ввести команду:

```
users
```

Если известно имя пользователя и требуется узнать о нём больше информации, необязательно читать /etc/passwd. Всё то же самое можно посмотреть с помощью команды `pinky -l`:

```
pinky -l test-user
```

В качестве альтернативы можно использовать команду `w` она тоже показывает всех авторизованных в текущий момент пользователей, включая создаваемую ими нагрузку на сервер.

Узнать UID пользователя можно с помощью команды `id`:

```
id username
```

Вывод команды будет похожим на следующий:

```
uid=1001(username) gid=1001(username)
groups=1001(username),27(sudo)
```

В выводе содержится информация о том, что учетная запись username существует, она находится в основной группе username и дополнительной группе `sudo`; идентификатор учетной записи – 1001.

Также можно получить строку о пользователе из файла /etc/passwd:

```
getent passwd username
```

Вывод команды будет похожим на следующий:

```
username:x:1001:1001::/home/username:/bin/sh
```

В данном выводе пользователь username имеет идентификатор пользователя и группы 1001, его домашняя директория – /home/username, командная оболочка – /bin/sh.

Для именования пользователей в ОС есть набор стандартных правил:

– имя пользователя может содержать только английские буквы [a-z] в верхнем и нижнем регистре, цифры, символ "_", тире "-" и точку;

– имя пользователя может оканчиваться символом "$";

– имя пользователя не может начинаться с тире, содержать только цифры или состоять из "." или "..";

– не рекомендуется использовать точку в начале имени пользователя;

– имя пользователя может включать до 32 символов.

3.2.2 Списки пользователей

Посмотреть список пользователей можно в файле /etc/passwd:

```
cat /etc/passwd
```

Вывод команды будет похож на следующий:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

Как правило, большая часть данных пользователей является системными и их UID меньше "1000" и больше "60000".

Более функциональная команда для отображения содержимого файла passwd getent:

```
getent passwd
```

Можно найти пользователя по идентификатору с помощью команды:

```
getent passwd 1000
```

Можно получить список несистемных пользователей с помощью команды:

```
getent passwd {1000..60000}
```

Получить только список логинов несистемных учетных записей можно с помощью команды:

```
getent passwd {1000..60000} | awk -F: '{ print $1}'
```

3.2.3 Использование sudo и привилегии

Разделение привилегий является одной из основных мер безопасности в ОС. Обычные пользователи работают с ограниченными правами и могут изменять только собственную рабочую среду. Для выполнения административных действий используется учётная запись суперпользователя root.

Вход в Систему от имени root по умолчанию заблокирован. Для повышения полномочий до административных рекомендуется использовать команду sudo. Первый созданный в Системе пользователь-администратор автоматически получает право на использование этой команды.

Команда sudo (расшифровывается как superuser do) позволяет временно получить права суперпользователя для выполнения одной команды. Такой подход обеспечивает дополнительный уровень безопасности, так как пользователи не всегда работают под учётной записью суперпользователя root.

Пример базового использования:

```
sudo команда
```

После ввода команды с префиксом sudo Система запросит пароль пользователя. Если пользователь входит в группу, имеющую соответствующие права, команда будет выполнена с административными полномочиями.

3.2.3.1 Конфигурация sudo

Настройка прав доступа осуществляется через файл /etc/sudoers. Он определяет, какие пользователи или группы могут использовать sudo и какие команды доступны. Для редактирования применяется утилита visudo, которая проверяет синтаксис и предотвращает ошибки при сохранении.

Пример типового содержимого файла /etc/sudoers:

```
Defaults    env_reset
Defaults    mail_badpass
Defaults
secure_PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/snap/bin"

root        ALL=(ALL:ALL) ALL
%admin      ALL=(ALL) ALL
%sudo       ALL=(ALL:ALL) ALL

#includedir /etc/sudoers.d
```

Основные элементы файла /etc/sudoers:

– Defaults env_reset — сброс переменных окружения для повышения безопасности;

– Defaults mail_badpass — уведомление о неудачных попытках ввода пароля;

– Defaults secure_PATH — безопасный список путей для выполнения команд через sudo.

Правила доступа в файле /etc/sudoers:

– root ALL=(ALL:ALL) ALL — пользователь root может выполнять любые команды от имени любых пользователей и групп.

– %admin ALL=(ALL) ALL и %sudo ALL=(ALL:ALL) ALL — аналогичные права назначаются группам admin и sudo.

Строка #includedir /etc/sudoers.d указывает на каталог, содержащий дополнительные правила. Файлы в этом каталоге обрабатываются так же, как и основной файл /etc/sudoers. Это позволяет приложениям добавлять собственные правила и облегчает администрирование. Редактирование этих файлов также выполняется через visudo:

```
sudo visudo -f /etc/sudoers.d/имя_файла
```

3.2.3.2 Присвоение привилегий

Для предоставления пользователю прав администратора его необходимо добавить в группу wheel (по умолчанию группа администраторов в ОС) с помощью одной из следующих команд:

```
usermod -aG wheel username
```

или

```
gpasswd -a username wheel
```

Если в ОС добавление пользователя в группу не срабатывает сразу же, может потребоваться редактирование файла /etc/sudoers, чтобы убрать символ комментария перед именем группы:

```
sudo visudo
/etc/sudoers
...
%wheel ALL=(ALL) ALL
...
```

3.2.3.3 Создание правил в файле sudoers

Файл /etc/sudoers позволяет гибко настраивать права пользователей при работе с sudo. Помимо стандартных строк, допускается использование псевдонимов и специальных свойств, которые обеспечивают удобство управления и повышенную безопасность.

3.2.3.3.1 Группировка пользователей и команд

Для объединения пользователей или команд применяются псевдонимы.

Например, можно создать три разных группы пользователей с некоторыми общими участниками:

```
...
User_Alias          GROUPONE = abby, brent, carl
User_Alias          GROUPTWO = brent, doris, eric,
User_Alias          GROUPTHREE = doris, felicia, grant
...
```

где GROUPONE, GROUPTWO и GROUPTWO – имена групп.

Имена групп должны начинаться с заглавной буквы. Затем можно дать участникам группы GROUPTWO разрешение на обновление базы данных dnf, создав следующее правило:

```
...
GROUPTWO  ALL = /usr/bin/dnf update
...
```

Если не указать пользователя или группу для запуска, команда `sudo` по умолчанию использует пользователя root.

3.2.3.3.2 Свойства правил

– Для управления поведением `sudo` используются дополнительные свойства.

– **NOPASSWD** — выполнение команды без запроса пароля:

```
GROUPONE ALL = NOPASSWD: /usr/bin/updatedb
```

– **PASSWD** — выполнение команды с обязательным вводом пароля (значение по умолчанию):

```
GROUPTWO ALL = PASSWD: /bin/kill
```

– **NOEXEC** — запрет выполнения вложенных команд внутри некоторых программ.

Это свойство полезно для предотвращения запуска других утилит через оболочку программы, например, less:

```
username ALL = NOEXEC: /usr/bin/less
```

В ходе практической работы с присвоением привилегий рекомендуется:

– ограничивать права пользователей только необходимыми командами;

– для систем с несколькими администраторами удобно использовать псевдонимы и вынесение правил в каталог /etc/sudoers.d;

– все изменения следует выполнять только через visudo, чтобы исключить ошибки синтаксиса.

3.2.3.4 Журналирование и безопасность

Все действия, выполненные через sudo, записываются в системный журнал. Записи можно просмотреть в файлах /var/log/auth.log или /var/log/secure. Это позволяет отслеживать действия пользователей с административными правами.

Рекомендуется:

– не использовать sudo для команд, не требующих повышенных привилегий;

– проверять корректность команд перед выполнением;

– ограничивать круг пользователей, имеющих доступ к sudo;

– назначать точечные правила для выполнения только необходимых команд.

Подробное описание доступно в справочном руководстве по команде:

```
man sudo
```

3.2.4 Управление учетными записями пользователей

Для управления учётными записями (УЗ) в Системе используются следующие утилиты:

– whoami — вывод имени текущего пользователя;
– who — просмотр активных пользователей;
– w — просмотр пользователей и их процессов;
– last — просмотр последних входов.

Основные операции по работе с пользователями включают создание и удаление учётных записей, изменение параметров, управление сроком действия пароля и блокировку.

3.2.4.1 Создание пользователя (useradd)

Для добавления новой учётной записи используется команда useradd.

Примеры использования команды useradd (где test-user – имя учетной записи пользователя):

– создание пользователя и домашнего каталога:

```
useradd -m test-user
```

– создание пользователя и добавление его в группу:

```
useradd -G printer -m test-user
```

– создание пользователя с возможностью получения привилегий суперпользователя:

```
useradd -G wheel -m test-user
```

– создание пользователя с определёнными UID и GID (группа с GID должна быть создана заранее):

```
useradd -u 900 -g 950 test-user
```

– создание пользователя с указанием домашней директории:

```
useradd -d /home/newdmosk test-user
```

– создание пользователя без возможности входа в Систему:

```
useradd -s /sbin/nologin test-user
```

После создания учётной записи пароль назначается командой:

```
passwd test-user
```

3.2.4.2 Изменение параметров пользователя (usermod)

Для изменения параметров учётной записи применяется команда `usermod`.

Примеры использования `usermod` (где test-user – имя учетной записи пользователя):

– смена основной группы:

```
usermod -g newgroup test-user
```

– добавление пользователя в дополнительную группу:

```
usermod -a -G group test-user
```

– требование сменить пароль при следующем входе:

```
chage -d 0 test-user
```

3.2.4.3 Удаление пользователя (userdel)

Удаление учётной записи выполняется командой `userdel` (где test-user – имя учетной записи пользователя):

```
userdel test-user
```

При необходимости удаления также домашнего каталога используется параметр -r:

```
userdel -r test-user
```

3.2.4.4 Управление сроком действия учётной записи (chage)

Команда `chage` применяется для управления сроками действия пароля и учётной записи.

Примеры использования команды `chage` (где test-user – имя учетной записи пользователя):

– автоматическая блокировка учётной записи с определённой даты:

```
chage -E 2025-05-01 test-user
```

– просмотр параметров:

```
chage -l test-user
```

– бессрочная учётная запись:

```
chage -E -1 test-user
```

– настройка максимального срока действия пароля (например, 30 дней):

```
chage -M 30 test-user
```

3.2.4.5 Управление пользовательскими сессиями

Для контроля текущих сеансов используются:

– просмотр списка активных пользователей с помощью команды:

```
who
```

– завершение сессии пользователя (где test-user – имя учетной записи пользователя):

```
pkill -KILL -u test-user
```

3.2.4.6 Контроль неудачных попыток входа (faillog)

Команда `faillog` отображает информацию о неудачных попытках входа и позволяет управлять ограничениями.

Примеры:

– просмотр записей о неудачных входах:

```
faillog
```

– обнуление счётчика попыток (где test-user – имя учетной записи пользователя):

```
faillog -r -u test-user
```

Записи о попытках хранятся в /var/log/faillog. По умолчанию для учётной записи действует лимит в 10 попыток входа.

3.2.5 Принудительное завершение сеанса пользователя

ОС предусматривает механизм контроля времени бездействия пользователя с последующим принудительным завершением графического сеанса. Функционал реализуется с помощью сценария /usr/bin/gui-kick, вызываемого по расписанию через механизм systemd (unit- и timer-файлы). О настройках timer-файлов подробнее см. раздел 3.4.

Сценарий /usr/bin/gui-kick использует утилиту xprintidle, определяющую длительность бездействия пользователя в графическом окружении в миллисекундах. На основании возвращаемого значения сценарий сравнивает время бездействия с двумя порогами:

– порог предупреждения — при превышении формируется уведомление о скором завершении сеанса;
– порог завершения сеанса — при достижении которого сценарий инициирует принудительное завершение графического сеанса.

Уведомление отображается через механизм notify-send в активном графическом сеансе пользователя.

Для периодического запуска сценария используется пара файлов:

– unit-файл службы /etc/systemd/system/gui-kick.service;
– таймер /etc/systemd/system/gui-kick.timer.

Пример типовой конфигурации файлов:

– gui-kick.service:

```
[Unit]
Description=GUI idle session termination check

[Service]
Type=simple
ExecStart=/usr/bin/gui-kick
```

– gui-kick.timer:

```
[Unit]
Description=Periodic GUI idle session check
```

```
[Timer]
OnCalendar=*-*-* *:*:00
AccuracySec=1s

[Install]
WantedBy=timers.target
```

Строчка OnCalendar=*-*-* *:*:00: означает, что таймер запускает сценарий каждую минуту, и где:

– *-*-* — любая дата (год-месяц-день);
– *:*: — любой час и любая минута;
– 00 — запуск в нулевой секунде каждой минуты.

Для включения механизма выполняется активация timer-файлов командами:

```
systemctl enable gui-kick.timer
systemctl start gui-kick.timer
```

Статус работы timer-файла можно проверить командами:

```
systemctl status gui-kick.timer
systemctl list-timers
```

Интервал бездействия задаётся переменной TIMEOUT, определённой в сценарии /usr/bin/gui-kick и выраженной в секундах. Изменение значения переменной позволяет адаптировать механизм под пользовательские требования.

Сценарий предусматривает следующий порядок действий:

– за 60 секунд до завершения сеанса пользователю отправляется предупреждение о необходимости проявить активность;
– если в течение указанного интервала активность не обнаружена, сценарий завершает графический сеанс принудительно, без учета состояния приложений и открытых окон.

3.2.6 Восстановление забытого пароля пользователя

В случае утраты пароля к локальной учётной записи, включая пользователя root, восстановление доступа возможно с использованием Live-образа дистрибутива. Для восстановления пароля пользователя в ОС пользователю необходимо выполнить следующие шаги:

а) Подготовить загрузочный носитель:

1) загрузить ISO-образ дистрибутива ОС (или совместимого Linux-дистрибутива);

2) записать образ на USB-накопитель или DVD;

3) загрузить Системы в режиме Live (без установки на диск).

б) Для определения раздела с установленной Системой выполнить следующие действия:

1) открыть программу GParted (в Live-образах ОС РОСА "ХРОМ" она включена по умолчанию, подробнее о работе с программой см. раздел 3.7.4). При использовании другого дистрибутива установить GParted по необходимости;

2) определить раздел с установленной ОС – раздел с точкой монтирования, например, /dev/sda2 (рисунок 11).

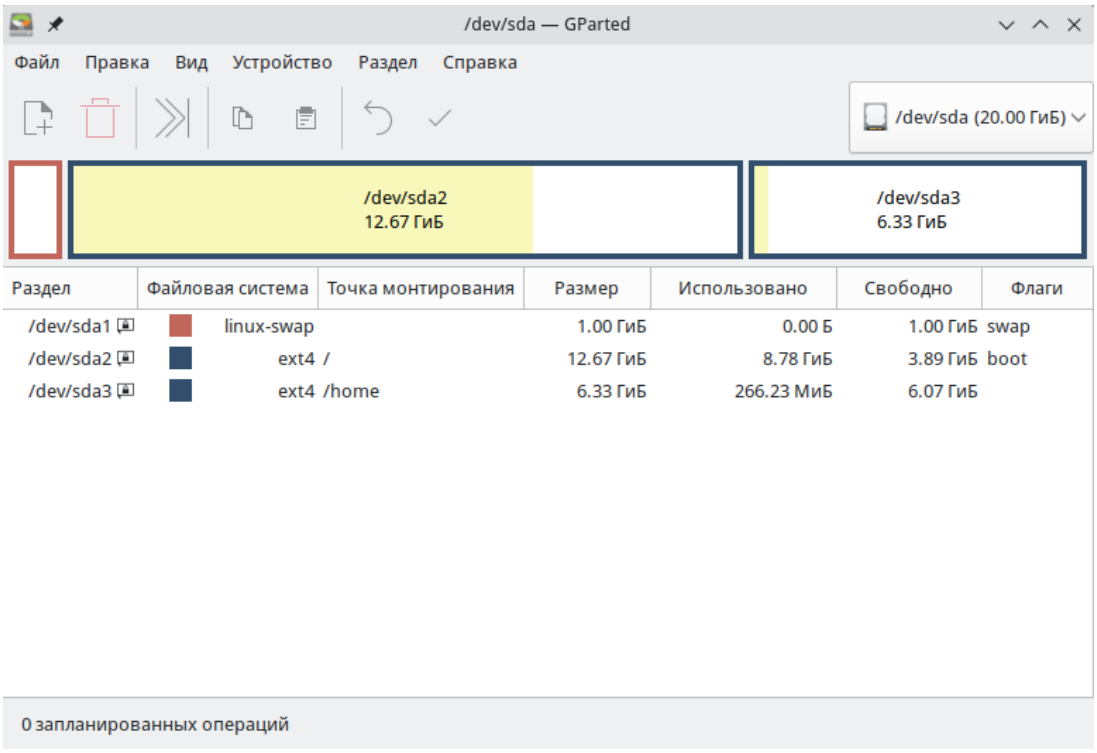


Рисунок 11 – Интерфейс программы GParted

в) Произвести монтирование установленной Системы и войти в chroot:

1) открыть терминал;

2) создать директорию для монтирования:

```
sudo mkdir /chroot
```

3) смонтировать корневой раздел установленной Системы:

```
sudo mount /dev/sdXY /chroot
```

При необходимости заменить sdXY на имя раздела с точкой монтирования, определённое ранее (например, sda2);

4) перейти в окружение установленной Системы с помощью chroot:

```
sudo chroot /chroot
```

г) Сбросить пароль:

– для смены пароля пользователя root использовать команду:

```
passwd
```

– в запросах терминала необходимо ввести и подтвердить новый пароль.

– для смены пароля другого пользователя использовать команду:

```
passwd <имя_пользователя>
```

Если имя пользователя неизвестно, его можно определить одной из следующих команд:

– просмотр содержимого директории /home;
– вывод всех пользователей:

```
cut -d: -f1 /etc/passwd
```

д) Завершить работу:

1) завершить сеанс chroot, выйти из Live-режима и перезагрузить компьютер;

2) при следующей загрузке войти в Систему с новым паролем.

3.2.7 Группы пользователей

Информация о группах пользователей хранится в файле /etc/group.

Группы применяются для делегирования прав доступа на определённые файлы, папки, скрипты сразу нескольким пользователям.

В качестве примера можно рассмотреть работу группы пользователей с FTP-сервером. Для работы с файлами нужно выбрать какую-либо директорию, создать группу пользователей и присвоить группе выбранную директорию. При использовании группы не нужно отдельно настраивать права каждому новому пользователю: достаточно добавить его в эту группу, и он автоматически получит доступ к FTP-каталогу.

3.2.7.1 Создание групп

Для создания групп используется команда `groupadd`:

```
groupadd new-group
```

Параметры команды приведены ниже (таблица 2).

Таблица 2 – Параметры команды groupadd

Параметр	Значение
-f	Если группа с указанным именем или GID уже существует, опция прерывает выполнение команды без соответствующей ошибки
-g 100500	Позволяет назначить свой GID для создаваемой группы
-r	Создает системную группу
-p p@ssw0rd	Устанавливает для группы пароль p@ssw0rd. Пароль запрашивается системой при попытке входа в группу с помощью команды `newgrp`. Не рекомендуется к использованию из-за проблем с безопасностью. Настроенный таким образом пароль можно увидеть в истории команд.

3.2.7.2 Редактирование групп пользователей

Для редактирования групп используется команда `groupmod`. Список изменений задаётся с помощью параметров, перечисленных ниже (таблица 3).

Таблица 3 – Параметры команды `groupmod`

Параметр	**Значение**
-g 100500	Меняет GID группы на 100500
-n another-name	Меняет имя группы на another-name

Например, если нужно изменить имя группы test-group на имя named-group, команда будет выглядеть так:

```
groupmod -n named-group test-group
```

3.2.7.3 Удаление групп пользователей

Нельзя удалить группу, если она указана в качестве основной для какого-либо существующего пользователя. Предварительно необходимо удалить этого пользователя из группы.

Непосредственно удаление группы выполняется одной командой:

```
groupdel test-group
```

Как и в случае удаления пользователей, необходимо вручную проверить, что на сервере не осталось данных, принадлежащих удалённой группе.

3.2.7.4 Управление группами

Базовым инструментом для управления группами является утилита gpasswd. Большинство параметров команды (кроме -A и -M) не сочетаются между собой, то есть в команде может быть только один параметр при разовом выполнении.

Общий вид команды:

```
gpasswd [что сделать] [в какой группе]
```

Параметры команды приведены ниже (таблица 4).

Таблица 4 - Параметры команды `gpasswd`

Параметр	Значение
-a new-user	Добавляет пользователя new-user в группу
-d bad-user	Удаляет пользователя bad-user из группы
-A user1,user2,...	Доступна для использования привилегированным пользователям (с правами root). Назначает список пользователей-администраторов группы
-M user1,user2,...	Доступна для использования привилегированным пользователям. Назначает список участников группы
-r	Отключает пароль группы. После этого только члены группы смогут использовать команду `newgrp` для подключения к группе
-R	Отключает внешний доступ к группе. После этого только члены группы смогут использовать команду `newgrp` для подключения к группе

Если потребуется добавить пользователя в новую группу, достаточно будет использовать следующую команду:

```
gpasswd -a new-user test-group
```

Также для добавления пользователей в новую группу используется описанная выше команда `usermod`. В следующем примере пользователь test-user добавляется в группу new-group:

```
usermod -a -G new-group test-user
```

Если нужно указать группу new-group в качестве основной группы пользователя test-user, то используется следующая команда:

```
usermod -g new-group test-user
```

Помимо этого, любой пользователь может сам авторизоваться и добавиться в новую группу с помощью команды:

```
newgrp new-group
```

Эта команда позволяет переключить группу пользователя в рамках текущей сессии, а также автоматически добавляет запрошенную группу в список групп пользователя.

Для определения, в каких группах состоит пользователь, необходимо воспользоваться командой `groups`:

```
user@rosa:~$ groups

user adm cdrom sudo dip plugdev lxd lpadmin sambaShare
```

Из этого примера видно, что пользователь user состоит в группах user и т.д. Если требуется посмотреть, в каких группах состоит другой пользователь, то нужно передать его имя в качестве аргумента:

```
user@rosa:~$ groups root

root : root
```

3.3 Команды

Команды можно выполнять с помощью командной строки, указав имя файла (бинарного файла или сценария).

Каждая запущенная команда является отдельным процессом. Если команда выполняется на переднем плане (по умолчанию все команды выполняются именно так), то пользователь не сможет работать с терминалом, пока команда не будет выполнена.

Важно отметить, что в ОС почти всегда учитывается регистр, включая имена файлов и каталогов, команды, аргументы и опции. Если что-то не работает, следует проверить написание команды.

Некоторые ключевые команды, необходимые для администрирования и работы с Системой.

Информация о Системе:

– `uname -a` — выводит информацию о ядре и Системе;
– `hostnamectl` — показ информации о хосте и Системе;
– `lscpu` — отображение информации о процессоре;
– `lsblk` — показ подключенных дисков и их разделов.

Управление пакетами:

– `sudo dnf install <пакет>` — установка пакетов с помощью пакетного менеджера dnf;
– `sudo dnf update` — обновление Системы;
– `sudo dnf remove <пакет>` — удаление пакета;
– `rpm -q <пакет>` — проверка установленного пакета.

Управление сервисами (systemd):

– `sudo systemctl start <сервис>` — запуск сервиса;
– `sudo systemctl stop <сервис>` — остановка сервиса;
– `sudo systemctl restart <сервис>` — перезапуск сервиса;
– `sudo systemctl status <сервис>` — проверка статуса сервиса;
– `sudo systemctl enable <сервис>` — включение автозапуска сервиса при старте Системы;
– `sudo systemctl disable <сервис>` — отключение автозапуска сервиса.

Работа с файлами и каталогами:

– `ls -la` — просмотр содержимого каталога с подробной информацией;
– `cp <файл> <путь>` — копирование файла;
– `mv <файл> <путь>` — перемещение или переименование файла;
– `rm <файл>` — удаление файла;
– `mkdir <каталог>` — создание нового каталога;
– `rmdir <каталог>` — удаление пустого каталога.

Мониторинг Системы:

– `top` или `htop` — отображение активных процессов и использование ресурсов;
– `df -h` — информация о свободном и занятом месте на дисках;
– `du -sh <каталог>` — подсчет занимаемого места каталогом;
– `free -h` — показ использования оперативной памяти.

Сетевые команды:

– `ip addr` — информация о сетевых интерфейсах;
– `ping <хост>` — проверка доступности удаленного узла;
– `traceroute <хост>` — маршрут до удаленного узла;

– `netstat -tuln или ss -tuln` — показ открытых портов и прослушиваемых сервисов.

Работа с файлами и архивами:

– `tar -cvf <архив.tar> <папка>` — архивация папки;
– `tar -xvf <архив.tar>` — извлечение содержимого архива;
– `gzip <файл>` — сжатие файла;
– `gunzip <файл.gz>` — распаковка файла.

Управление пользователями и правами:

– `sudo useradd <имя>` — создание нового пользователя;
– `sudo passwd <имя>` — изменение пароля пользователя;
– `sudo usermod -ag <группа> <имя>` — добавление пользователя в группу;
– `sudo chown <пользователь>:<группа> <файл>` — изменение владельца и группы файла;
– `sudo chmod 755 <файл>` — изменение прав доступа к файлу.

Журналы:

– `journalctl -xe` — просмотр системных журналов;
– `journalctl` — просмотр всех записей журнала с момента последней загрузки Системы;
– `journalctl -f` — просмотр последних записей в реальном времени;
– `dmesg` — журнал ядра (информация о загрузке и драйверах).

Полный список доступных консольных команд приведен в Приложении А настоящего руководства.

3.3.1 Текстовые редакторы

3.3.1.1 Текстовый редактор vi/vim

Текстовый редактор vi является базовым редактором, входящим в состав большинства Unix-подобных ОС и доступным в минимальных и аварийных средах. Редактор vim (Vi IMproved) представляет собой расширенную версию vi, включающую дополнительные возможности, улучшенный интерфейс и расширенную функциональность. Vim полностью совместим с vi и может работать в режиме, имитирующем поведение оригинального редактора.

Редакторы vi и vim относятся к модальным редакторам: управление текстом осуществляется в нескольких режимах, в каждом из которых выполняется ограниченный набор команд. Основными режимами являются режим команд (нормальный режим) и режим вставки.

Запуск редактора выполняется указанием имени файла:

```
vi имя_файла
vim имя_файла
```

Если указанный файл отсутствует, он будет создан при сохранении.

Для открытия файла только для чтения используется параметр:

```
vi -R имя_файла
```

Редакторы vi и vim используют два основных режима работы:

– режим команд (нормальный режим) — используется для выполнения команд перемещения, редактирования, удаления, копирования и навигации по тексту;

– режим вставки — используется для ввода текста; переход в этот режим выполняется нажатием клавиши `i`, `a`, `o` и других команд вставки.

Возврат в режим команд выполняется нажатием клавиши `Esc`.

Наиболее часто используемые команды редакторов:

– переход в режим вставки:

  – i — вставка в текущей позиции;
  – a — вставка после текущего символа;
  – o — вставка новой строки ниже;

– сохранение и выход:

  – :w — сохранить изменения;
  – :q — выйти из редактора;
  – :wq — сохранить и выйти;
  – :q! — выйти без сохранения;

– навигация:

  – клавиши h, j, k, l — перемещение влево, вниз, вверх, вправо;
  – gg — переход в начало файла;
  – G — переход в конец файла;

– удаление и редактирование:

  – x — удаление символа;
  – dd — удаление строки;
  – yy — копирование строки;
  – p — вставка скопированного фрагмента.

Редактор vim предоставляет ряд функций, отсутствующих в vi:

– подсветка синтаксиса для различных типов файлов;

– поддержка кодировок, включая UTF-8;
– расширенный механизм отмены и восстановления действий;
– многооконный режим (:split, :vsplit);
– поддержка плагинов и скриптов на языке vimscript;
– режим совместимости с vi (vim -v).

Редактор vi включён в состав минимального набора инструментов ОС и доступен в аварийном режиме, режиме восстановления и при загрузке из окружения chroot. Это позволяет выполнять редактирование критически важных конфигурационных файлов при отсутствии графического интерфейса или полноценных редакторов.

3.3.1.2 Текстовый редактор nano

Текстовый редактор nano является консольным редактором, ориентированным на пользователя и отличающимся простым интерфейсом и невысоким порогом освоения. В отличие от модальных редакторов vi и vim работа в nano не требует переключения между режимами, а управление осуществляется с помощью комбинаций клавиш и линейки подсказок, отображаемой в нижней части окна. Редактор широко применяется для редактирования конфигурационных файлов и оперативного изменения текстовой информации в терминале.

Запуск редактора выполняется командой:

```
nano имя_файла
```

Если указанный файл отсутствует, редактор создаст его при сохранении.

Для открытия файла в режиме только чтения используется параметр:

```
nano -v имя_файла
```

При запуске редактор отображает содержимое файла и две панели подсказок в нижней части экрана. Символ "^" в подсказках означает использование клавиши `Ctrl`.

Примеры сочетаний клавиш:

– `Ctrl` + `O` — сохранить файл;
– `Ctrl` + `X` — выйти из редактора;
– `Ctrl` + `G` — открыть встроенную справку;
– `Ctrl` + `K` — вырезать строку;
– `Ctrl` + `U` — вставить ранее вырезанный фрагмент;
– `Ctrl` + `W` — поиск текста;
– `Ctrl` + `\` — замена текста.

Для перемещения по тексту используются клавиши стрелок, клавиши `Page Up` и `Page Down`.

Редактирование в nano выполняется непосредственно при вводе текста. Для выполнения основных операций используются следующие сочетания клавиш:

– удаление символов — клавиша `Backspace` или `Delete`;
– удаление строки — `Ctrl` + `K`;
– копирование и вставка:

– копирование строки: `Alt` + `6` или `Esc`, затем `6`;
– вставка: `Ctrl` + `U`.

Редактор поддерживает работу с несколькими буферами, автоматический перенос строк и подсветку выбранных текстовых фрагментов.

Для выполнения поиска используется сочетание `Ctrl` + `W`.

После ввода искомого текста осуществляется переход к следующему совпадению клавишей `Ctrl` + `W`, затем `Enter`.

Функция замены вызывается сочетанием `Ctrl` + `\`.

Редактор последовательно предлагает заменить найденные совпадения или пропустить их.

Поведение редактора nano может быть настроено с использованием файла конфигурации /etc/nanorc.

В файле могут быть активированы следующие параметры:

– подсветка синтаксиса (include /usr/share/nano/*.nanorc);
– нумерация строк (set linenumbers);
– автоотступы (set autoindent);
– перенос строк (set softwrap);
– отображение позиции курсора (set cursorpos).

Дополнительные настройки могут задаваться в конфигурации пользователя ~/.nanorc.

Простота интерфейса делает редактор nano предпочтительным для администраторов в ситуациях, когда требуется оперативно изменить конфигурационный файл без необходимости использования модальных команд.

3.3.2 Выполнение команд

3.3.2.1 Без опций и аргументов

Чтобы запустить команду без аргументов и опций, нужно просто ввести саму команду и нажать клавишу `ENTER`.

В таком случае команда выполнит своё стандартное действие. У каждой команды индивидуальное поведение по умолчанию. Например, команда `cd` без аргументов откроет домашний каталог текущего пользователя. Команда `ls` без аргументов выведет список файлов и каталогов, хранящихся в текущем каталоге.

3.3.2.2 С аргументами

Многие команды принимают аргументы или параметры, которые могут влиять на их поведение. Например, в качестве аргумента команды `cd` часто используется имя каталога, в который нужно перейти. Например, чтобы перейти в /usr/bin, нужно ввести:

```
cd /usr/bin
```

Сначала нужно ввести саму команду, а после неё указать аргументы. Перейдя в новый каталог, путь в командной строке изменится.

3.3.2.3 С опциями

Большинство команд имеют список опций или флагов, которые позволяют изменять поведение команды.

Опции указываются после команды через дефис ("-"). Они бывают верхнего и нижнего регистра. Некоторые опции начинаются с двух дефисов ("--"). После одного дефиса можно указать несколько опций подряд, а после двух дефисов обычно идёт одна опция, состоящая из нескольких символов (обычно это описательное слово).

В качестве примера можно привести некоторые опции команды:

– `-l` – включает подробный листинг, который предоставляет дополнительные детали (права доступа, права собственности, размер файлов и т.п.).

– `-a` – выводит полный список файлов, которые хранятся в каталоге (включая скрытые файлы).

Флаг -l команде `ls` передается так:

```
ls -l
```

Такая команда выведет много дополнительной информации о файлах.

Опции можно группировать, например, передать команде `ls` флаги -l и –a одновременно:

```
ls -la
```

Эта команда выведет подробные данные обо всех файлах, включая скрытые.

3.3.2.4 С опциями и аргументами

Опции и аргументы почти всегда можно комбинировать при запуске команд.

Например, чтобы просмотреть полное содержимое каталога /home, можно использовать:

```
ls -la /home
```

Такая команда вернёт подробные данные о каждом файле заданного каталога.

3.3.3 Фоновые процессы

Обычно для запуска команды необходимо ввести ее имя. В командной строке можно указать сколько угодно специальных символов, которые будут интерпретированы bash еще до запуска команды. Таким же образом можно запускать команды в фоновом режиме.

Важнейший и самый востребованный подстановочный символ "&". Если указать его в конце командной строки, bash запустит программу в фоновом режиме. Это имеет смысл в первую очередь в случае, если на выполнение команды уходит много времени, чтобы можно было продолжать работу, не дожидаясь окончания выполнения такой программы.

```
find / -name '*sh' > result &
[1] 3345
```

Вышеуказанная команда ищет по всей файловой системе файлы, названия которых оканчиваются на "sh". Список найденных файлов записывается в файле result. Поскольку команда выполняется в фоновом режиме, работу можно не прерывать. Вывод "[1] 3345" означает, что фоновый процесс имеет номер "PID 3345". Здесь PID означает идентификатор процесса (process ID). Номер PID интересен в том случае, если процесс был аварийно завершен командой `kill`. Номер в квадратных скобках – это номер фонового процесса, запущенного в bash. Как правило, этот номер не важен.

Если при запуске команды не поставлен символ "&", не следует ни дожидаться окончания выполнения программы, ни принудительно завершать программу нажатием комбинации клавиш `Ctrl+C`. Гораздо лучше приостановить выполнение программы, нажав комбинацию клавиш `Ctrl+Z`, а затем продолжить ее работу в виде фонового процесса с помощью команды `bg`.

3.3.4 Выполнение нескольких команд

ОС позволяет производить запуск нескольких команд в одной строчке. Существует два способа:

- использование символа ";":

```
sudo dnf update; sudo dnf upgrade -y
```

Вторая команда будет запущена, даже если первая команда не будет выполнена. Если такой вариант не подходит, рекомендуется использовать второй вариант.

- использование символов "&&":

```
sudo dnf update && sudo dnf upgrade -y
```

В этом случае будут выполняться обе версии команды, но вторая форма будет выполнять вторую команду только в том случае, если первая завершится успешно.

При использовании символа "&" также можно указать следующую команду. В таком случае первая команда будет выполняться в фоновом режиме, а вторая на первом плане. Например, команда `find` запускается в фоновом режиме, а `ls` одновременно выводит содержимое текущего каталога:

```
find / -name '*sh' > result & ls
```

Если вместо символа "&" поставить точку с запятой, то bash поочередно выполнит команды в фоновом режиме:

```
rls; date
```

Эта команда сначала отображает содержание текущего каталога, а затем выводит текущий файл.

Если необходимо перенаправить всю эту информацию в файл с помощью символа ">", то обе команды ставятся в круглых скобках. В таком случае они выполняются одной и той же оболочкой:

```
user$ (ls; date) > content
```

В файле content теперь находится список файлов, созданный `ls`, а также текущая дата (из команды date). Благодаря круглым скобкам обе команды

выполняются одной и той же оболочкой и выдают общий результат (как правило, ситуация иная: при запуске каждой новой команды активируется новая оболочка).

– использование символов "||":

Оператор || используется для выполнения второй команды только в том случае, если первая завершилась неуспешно (код завершения не равен "0"), например:

```
команда1 || команда2
```

Если `команда1` завершится успешно, то `команда2` не выполнится. В противном случае, если `команда1` завершится с ошибкой, `команда2` будет выполнена.

Эти операторы (&& и ||) предоставляют удобные способы создания цепочек команд с учетом их успешного или неуспешного завершения, что может быть полезным в скриптах и командной строке для управления логикой выполнения команд, то есть в зависимости от результата другой команды.

Например, команда `user$ команда1 && команда2` выполняет `команда1`. Только в случае, если эта команда была выполнена успешно (без ошибки, без выдачи "0" в качестве возвращаемого значения), выполняется `команда2`.

Ниже приведены все варианты использования двух команд (таблица 5).

Таблица 5 - Использование двух команд

Команда	Функция
`команда1; команда2`	Выполняет команду друг за другом
`команда1 && команда2`	Выполняет `команда2`, если `команда1` была выполнена успешно
`команда1 || команда2`	Выполняет `команда2`, если `команда1` была выполнена с ошибкой
`команда &`	Запускает `команду` в фоновом режиме
`команда1 & команда2`	Запускает `команда1` в фоновом режиме, `команда2` на виду
`(команда1 ; команда2)`	Выполняет обе `команды` в одной и той же оболочке

Другие возможности для создания условий и ветвления команд связаны с использованием оболочковой команды `if`, которая интересна администраторам, которые собираются заниматься программированием на языке оболочки.

3.3.5 Потоки и перенаправление

Потоки являются базовым механизмом обмена данными в Системе. С их помощью обеспечивается взаимодействие программ и передача информации между процессами и файлами. Администратор может управлять источниками и приёмниками данных, перенаправляя ввод, вывод и сообщения об ошибках, а также объединяя команды в конвейеры для последовательной обработки информации.

В Системе каждая программа использует стандартные потоки данных:

– stdin — стандартный ввод (обычно клавиатура или другой источник данных);

– stdout — стандартный вывод (результат выполнения программы, выводимый в терминал);

– stderr — поток ошибок (вывод сообщений об ошибках).

Для управления потоками применяется механизм перенаправления:

– > — вывод в файл (с перезаписью существующего содержимого);
– >> — вывод в файл с добавлением в конец;
– < — ввод из файла вместо клавиатуры;
– 2> — перенаправление сообщений об ошибках в файл;
– &> или >& — объединение вывода и ошибок в один файл;
– | (конвейер) — передача вывода одной команды на вход другой;
– | tee файл — одновременный вывод в терминал и запись в файл.

Примеры работы с потоками:

```
    ls *.txt > content.txt          # сохранить список файлов в файл
    sort < input.txt > sorted.txt  # сортировка данных из файла и запись результата
```

Следует учитывать, что перенаправление в тот же файл, из которого ведётся чтение (sort dat > dat), приведёт к его повреждению.

В Системе потоки реализуются как часть процессов. Новый процесс наследует файловые дескрипторы от родительского, что обеспечивает единый механизм обмена данными между программами.

3.3.6 Поиск и фильтрация информации

В Системе предусмотрено множество возможностей поиска файлов. Выбор команды, лучше всего подходящей для конкретного случая, зависит от типа файла (текстовый файл, программа и т. д.) и его содержания (фрагменты названия, фрагменты текста и т. д.).

Ниже приведены инструменты для поиска файлов (таблица 6).

Таблица 6 - Команды поиска файлов

Команда	Функция
`grep`	Поиск текста в текстовом файле
`find`	Поиск файлов по имени, дате, размеру и т.д.
`locate`	Поиск фалов по имени
`whereis`	Поиск файлов заданных каталогах
`which`	Поиск программ в PATH-каталогах

3.3.6.1 Команды which, whereis, whatis

Команда `which` ищет указанную команду и возвращает ее полное имя, если оно было набрано без указания пути. Команда `which` осуществляет поиск только по каталогам, указанным в PATH, работая при этом исключительно быстро. PATH содержит список каталогов, в которых находятся программы. Однако необходимо учитывать, что PATH для администратора содержит больше каталогов, чем для обычного пользователя. Таким образом, для поиска системных команд необходимо войти в Систему с правами администратора.

```
user$ which emacs
/usr/bin/emacs
```

Команда `whereis` просматривает все пути, обычно используемые для двоичных файлов, файлов конфигурации, man-страниц и исходного кода. Для этого ее нужно ввести после имени файла. Команда `whereis` охватывает больше файлов, чем `which`, при этом поиск не ограничивается одними только программами. Она не может искать в каталогах, которые специально не указаны программе как места для поиска.

```
user$ whereis fstab
fstab: /etc/fstab /usr/include/fstab.h
```

Команда `whatis` выводит краткое однострочное описание из руководства (man).

```
# whatis ls cp rm mv
ls (1) list directory contents
cp (1) copy files and directories
rm (1) remove files or directories
mv (1) move (rename) files
```

Также можно использовать команду, которая выводит путь к исполняемому файлу (или команде) в командной строке:

```
command -V команда1
```

Например, следующая команда может быть полезна, чтобы узнать место загрузки определенной команды в Системе:

```
command -V ls
```

3.3.6.2 Команда locate

Команда `locate` ищет файлы, в полном названии которых (путь + имя файла) содержится указанная схема поиска. Поиск осуществляется очень быстро: `locate` не ищет по файловой системе, а обращается к БД, в которой содержится список имен всех файлов из файловой системы. Команда `locate` показывает все файлы, к которым пользователь может получить доступ. Если нужно найти системные файлы, то следует войти в Систему как администратор и выполнить команду `locate`.

Например, следующая команда ищет конфигурационный X-файл xorg.conf:

```
user$ locate xorg.conf
/etc/X11/xorg.conf
/etc/X11/xorg.conf.backup
```

3.3.6.3 Особенности

Внедрение команд `locate` и `updatedb` отличается от дистрибутива к дистрибутиву. В ОС РОСА "ХРОМ" эти команды входят в состав стандартного пакета mlocate. Файловая база данных находится в файле /var/lib/mlocate/mlocate.db и ежедневно обновляется программой Cron-Job /etc/cron.daily/mlocate. Конфигурационный файл /etc/updatedb.conf определяет, какие каталоги и файловые системы не будут учитываться при поиске (например, CD, DVD, различные буферные каталоги).

3.3.6.4 Команда wc

Команда `wc` позволяет выполнять подсчёт, считывая стандартный ввод или составной список и генерируя одну или несколько из следующих статистических данных: количество строк, количество слов и количество байтов.

Общий вид команды:

```
user$ wc [ПАРАМЕТР]… [ФАЙЛ]…
```

Параметры команды `wc` представлены ниже (таблица 7):

Таблица 7 - Параметры команды `wc`

Параметр	Значение
-c, --bytes	Вывести количество байтов
-m, --chars	Вывести количество символов
-l, --lines	Вывести количество новых строк
-w, --words	Вывести количество слов

Вывод команды `wc` представляется в следующем виде:

– первая колонка, количество строк (-l);
– вторая колонка, количество слов (-w);
– третья колонка, количество символов (-m).

3.3.6.5 Команда sort

Команда `sort` используется в ОС для сортировки содержимого текстовых файлов. Это позволяет выводить текстовые строки в определённом порядке.

Общий вид команды:

```
sort [ПАРАМЕТР]… [ФАЙЛ]…
```

Команда направляет сортированное слияние всех [ФАЙЛ] на стандартный вывод.

Если [ФАЙЛ] не задан или задан как "-", то команда читает стандартный ввод.

Параметры команды `sort` представлены ниже (таблица 8).

Таблица 8 - Параметры команды `sort`

Параметр	Значение
-b, --ignore-leading-blanks	Игнорировать начальные пропуски
-d, --dictionary-order	Рассматривать только пропуски, буквы и цифры
-f, --ignore-case	Игнорировать регистр букв

Параметр	Значение
-g, --general-numeric-sort	Сравнивать в соответствии с общим числовым значением
-i, --ignore-nonprinting	Рассматривать только печатные символы
-h, --human-numeric-sort	Сравнивать числа в читабельном виде (например, 2K 1G)
-n, --numeric-sort	Сравнивать по числовым значениям строк
-r, --reverse	Изменить результаты сравнения на противоположные
-c, --check, --check=diagnose-first	Проверять, сортированы ли входные файлы
-k, --key=ОПР_КЛЮЧА	Сортировать по ключу; в ОПР_КЛЮЧА задано расположение и тип
-m, --merge	Объединить уже отсортированные файлы
-o, --output=ФАЙЛ	Выводить в ФАЙЛ, а не на стандартный вывод
-u, --unique	С -c проверять порядок строго; без -c выводить только первое среди нескольких равных

Примечание – ОПР_КЛЮЧА задается как F[.C][OPTS][,F[.C][OPTS]] для начала и конца положения, где F – номер поля, а C – позиция в поле; оба отсчитываются от **1**, а конечная позиция по умолчанию равна концу строки.

Если не задан параметр -t или -b, то символы в поле отсчитываются от начала предшествующего пробельного символа. OPTS составляется из одной или нескольких букв, задающих порядок сортировки [bdfgiMhnRrV]; он переопределяет соответствующие глобальные параметры сортировки для данного ключевого поля.

Сортировка по умолчанию (без опций):

```
sort example.txt
```

Обратная сортировка:

```
sort -r example.txt
```

3.3.6.6 Команда cut

Команда `cut` предназначена для вырезания части текста из файла и записи результата в стандартный вывод.

Общий вид команды:

```
cut [опция] [файл]
```

Параметры команды `cut` представлены ниже (таблица 9).

Таблица 9 - Параметры команды `cut`

Параметр	Значение
-b, --bytes=СПИСОК	Выбрать только заданные байты
-c, --characters=СПИСОК	Выбрать только заданные символы
-d, --delimiter=РАЗДЕЛИТЕЛЬ	Использовать для разделения полей РАЗДЕЛИТЕЛЬ вместо табуляции
-f, --fields=СПИСОК	Выбрать только заданные поля; также печатать все строки, не содержащие разделителей, если только не задан параметр -s
--complement	Дополнить множество выбранных байтов, символов или полей
-s, --only-delimited	Не выводить строки, не содержащие разделителей
--output-delimiter=СТРОКА	Использовать СТРОКА для разделения полей при выводе, по умолчанию используется разделитель для ввода
-z, --zero-terminated	Разделитель строк NULL, а не символ новой строки

3.3.6.7 Команда find

Если не указывать дополнительные параметры, команда `find` выдает список всех файлов, находящихся в текущем каталоге и всех его подкаталогах:

Общий вид команды:

```
user$ find
```

Следующая команда ищет все файлы в текущем каталоге и во всех подкаталогах, названия которых начинаются с ".t":

```
user$ find -name '.e*'
./.thunderbird
./.thunar
```

```
./.textgen
…
```

В следующем примере команда `find` ищет все каталоги, находящиеся в /etc/. Обычные файлы, располагающиеся в /etc/, среди результатов не показываются. Список результатов упорядочивается по алфавиту с помощью команды `sort` (по умолчанию такой сортировки не происходит).

```
root# find /etc -type d | sort
/etc
/etc/alsa
/etc/alsa/conf.d
/etc/alternatives
...
```

В следующем примере команда `find` ищет все файлы в (под)каталогах /home, принадлежащих пользователям группы users, причем искомые файлы должны были каким-либо образом быть изменены в течение последних пяти дней (содержание, права доступа и т. д.):

```
root# find /home -group users -mtime -5
...
```

Команда `find -mtime +5` находит такие файлы, которые были изменены ранее, чем пять дней назад, а команда `-mtime 5` возвращает файлы, которые были изменены ровно пять дней назад. При этом во многих случаях "вчера" означает для команды `find` "24 и более часов от данного момента". Если вместо -mtime применить параметр -ctime, то точкой изменения времени будет считаться "время изменения индексного дескриптора". Эта точка изменяется и тогда, когда изменились лишь права доступа, а содержимое каталога осталось прежним.

Следующая команда удаляет все резервные копии, содержащиеся в данном каталоге и во всех подкаталогах. При этом команда `find` строит список всех сомнительных файлов и передает его команде `rm` через подстановку команды `($(команда))`:

```
user$ rm $(find. -name '*~')
```

Если речь идет об очень большом количестве файлов, то при выполнении вышеуказанной команды возникает ошибка: запись со всеми *~-файлами может получиться настолько длинной, что превысит размер командной строки. В таких случаях следует воспользоваться либо параметром -exec- команды `find`, либо командой `xargs`.

3.3.6.8 Команда grep

Команда `grep` ищет в текстовом файле соответствия для заданной поисковой схемы. В зависимости от того, какие параметры настроены в конкретном случае, команда может дополнительно показывать найденные фрагменты текста или просто сообщать, в каком количестве строк была найдена заданная поисковая схема. Поисковая схема является так называемым регулярным выражением.

Следующая команда просматривает все tex-файлы текущего каталога в поисках последовательности символов emacs. Список всех найденных строк (перед каждой из которых указывается имя файла) отображается на экране:

```
grep emacs *.tex
```

Команда `grep` определяет, как часто применяется функция arctan в указанных c-файлах.

```
grep -c arctan\(.*\) *.c
```

Команда `grep` с параметром -v возвращает в качестве результата все строки, в которых отсутствует заданный шаблон поиска.

В приведенном далее примере команда `grep` удаляет из configfile все строки, которые начинаются с символа решетки (#) (то есть все комментарии). Следующая команда `cat` дополнительно удаляет все пустые строки. Конечный результат сохраняется в файле nocomments.

```
grep -v '^#' configfile | cat -s > nocomments
```

Эта команда очень удобна, если всего несколько строк конфигурационного кода приходится на сотни или тысячи строк комментариев.

3.3.6.9 Комбинации команд find и grep

Команды `find` и `grep` можно комбинировать, чтобы выполнять расширенный поиск. В следующем примере команда `find` просматривает все файлы на предмет того, нет ли в них последовательности символов emacs. Если такая последовательность обнаруживается, то название файла выводится на экран. **Следует обратить внимание**, что нельзя указывать параметр –print перед –exec (в отличие от предыдущего примера, команда `grep emacs *.tex` учитывает все файлы с расширением *.tex, независимо от глубины вложения подкаталога, в котором они могут находиться).

```
find -name '*.tex' -type f -exec grep -q emacs {} \; -print
```

Следующая команда просматривает в текущем каталоге все файлы размером менее 10 Кбайт на предмет наличия в них регулярного выражения case.*in. Список найденных файлов сохраняется в файле result. Ограничение размера файла вводится для того, чтобы исключить из поиска двоичные файлы (обычно они гораздо больше 10 Кбайт).

```
user$ find -name '*' -maxdepth 1 -size -10k -exec grep -q \
> case.*in {} \; -print > result
```

3.3.7 Переменные окружения

Переменные окружения представляют собой значения, влияющие на выполнение команд и процессов в ОС. При запуске Системы они устанавливаются на основе конфигурационных файлов по умолчанию и определяют параметры работы пользовательской оболочки.

Для вывода всех переменных окружения используется команда:

```
env
```

Вывод этой команды обычно очень объёмен и отображает полный список установленных переменных, требуется найти запись PATH вида:

```
PATH=/root/.local/bin:/root/bin:/sbin:/bin:/usr/sbin:/usr/bi
n
```

Переменная PATH содержит перечень каталогов, в которых оболочка ищет исполняемые файлы при выполнении команд.

В оболочке bash переменные можно просмотреть также с помощью встроенной команды:

```
set
```

Эта команда выводит все настройки оболочки, включая переменные окружения.

Просмотр только переменных окружения:

```
set | grep "^VAR="
```

Для фильтрации вывода по определённой переменной применяется команда:

```
set | grep "^PATH="
```

Чтобы извлечь значение переменной, нужно указать перед её именем символ "$".

Примеры:

```
echo $PATH      # вывод текущего пути поиска программ
```

```
cd $HOME        # переход в домашний каталог пользователя
```

Если переменная не установлена, команда вернёт пустую строку.

Для установки значения переменной используется команда:

```
ПЕРЕМЕННАЯ=значение
```

Если переменная уже существует, её значение будет перезаписано.

Чтобы сделать переменную доступной для дочерних процессов, применяется команда `export`:

```
export ПЕРЕМЕННАЯ=значение
```

Пример добавления нового каталога в конец переменной PATH, если приложение установлено в /opt/app/bin:

```
export PATH=$PATH:/opt/app/bin
```

3.3.8 Справочные ресурсы по командам

Справочные страницы онлайн-руководства (так называемые man-страницы, поскольку их можно просмотреть с помощью команды `man`) составляют традиционную "интерактивную" документацию.

Интерактивное руководство содержит краткое описание отдельных команд, драйверов, форматов файлов и библиотечных функций.

3.3.8.1 Команда man

Команда `man` имеет следующий формат и вызывает справочную страницу из указанного раздела:

```
man команда
```

Также можно использовать другой вариант команды:

```
man раздел заголовок
```

Например, команда `man sync` вызывает справочную страницу для команды `sync`, а команда `man 2 sync` для системного вызова `sync`.

Команда вида:

```
man -k ключевое_слово
```

или

```
apropos ключевое_слово
```

выводит список справочных страниц, в строке пояснений, к которым имеется указанное ключевое слово.

```
man -k translate

objcopy (1) copy and translate object files
dcgettext (3) translate message
tr (1) translate or delete characters
snmptranslate (1) translate SNMP OID values into more useful information
tr (lp) translate characters
```

База данных ключевых слов может устаревать. При добавлении новых справочных страниц к ОС, возможно, придется перестроить (перекомпоновать) БД с помощью команды `makewhatis`.

В целях экономии места на диске ОС сжимает справочные страницы с помощью утилиты Gzip (команда `man` может очень быстро разархивировать их). Команда `man` поддерживает кеш отформатированных страниц в каталогах /var/cache/man или /usr/share/man, если соответствующие каталоги доступны для записи, но эти операции рискованны с точки зрения безопасности.

Команда `man` ищет страницы в ряде каталогов. В ОС выяснить путь поиска позволяет команда `manpath`:

```
manpath
/usr/local/man:/usr/local/Share/man:/usr/Share/man
```

Эта установка хранится в переменной среды MANPATH, и при необходимости ее можно изменить.

```
export MANPATH=/home/Share/localman:/usr/Share/man
```

3.3.8.2 Команда TLDR

В качестве альтернативы для получения справочной информации используют утилиту tldr (сокращение от "Too long; didn't read").

Утилита tldr доступна из репозитория ОС и устанавливается командой:

```
sudo dnf in tldr -y
```

Например, для получения tldr-справки по tar выполняется команда:

```
tldr tar
```

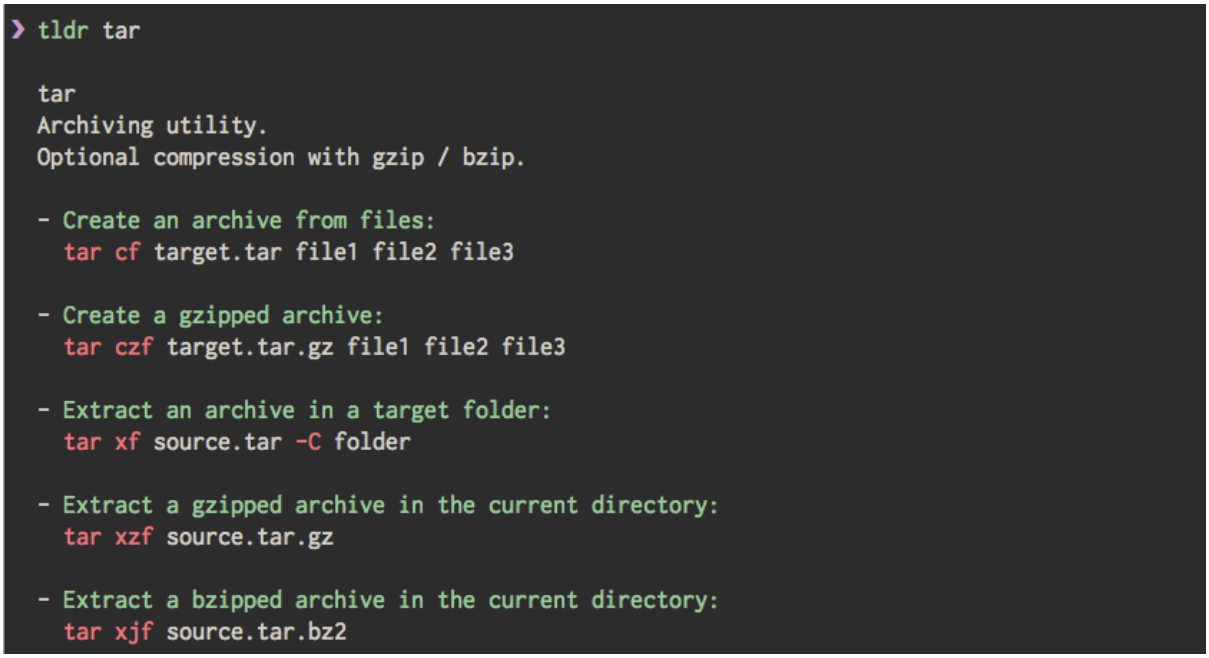


Рисунок 12 - Справка по tar

Вывод команды содержит минимальное описание назначения утилиты и компактный список из самых частых команд (рисунок 12).

3.3.8.3 Использование опций для получения справки

Многие консольные команды предоставляют подробную справочную информацию, которая может быть получена с использованием опции -h или --help. Эти опции позволяют пользователям быстро ознакомиться с основными параметрами и функциональностью команды без необходимости изучения руководства.

Эти опции обычно выводят краткую справочную информацию, включающую список доступных опций, их описание и примеры использования. Этот подход облегчает работу с новыми командами и помогает быстро освоить основные возможности.

Рекомендуется использовать опции -h или --help перед тем, как начать работу с новой командой, чтобы узнать больше о ее функциональности и опциях, которые могут быть полезны в конкретном контексте.

Синтаксис опции -h:

```
команда -h
```

Некоторые команды предоставляют дополнительную опцию --help для вывода справочной информации:

```
команда –help
```

3.4 Системный менеджер systemd

Systemd — это системный и сервисный менеджер, обеспечивающий инициализацию Системы, запуск и контроль служб, управление зависимостями и сбор журналов. Systemd применяется на всех этапах жизненного цикла службы: от автоматического запуска при старте до мониторинга состояния и перезапуска при сбоях.

Под "сервисами" в контексте Системы и systemd понимаются службы (daemon), то есть компоненты, выполняющиеся в фоновом режиме и предоставляющие функциональность Системе или приложениям.

Сервисы описываются юнитом типа .service и управляются по жизненному циклу: запуск, остановка, перезапуск, перезагрузка конфигурации, проверка состояния.

Основные задачи systemd:

– инициализация Системы при загрузке и параллельный запуск необходимых компонентов;

– управление службами: запуск, остановка, перезапуск, загрузка конфигурации, контроль зависимостей и порядка инициализации;

– мониторинг состояния: определение статусов active, inactive, failed и др., автоматический перезапуск по политике;

– журналирование и диагностика: сбор логов через компонент journald и предоставление доступа к журналам;

– управление зависимостями и целями: формирование целевых состояний (targets), гарантия корректной последовательности запуска.

Все команды управления службами systemd выполняются через утилиту systemctl.

Для диагностики применяется просмотр журналов через journald. Рекомендуется использовать подписанные и проверенные конфигурации юнитов и избегать ручного изменения файлов в /usr/lib/systemd/system, выполняя переопределения в /etc.

3.4.1 Управление службами

Команда `systemctl` является основным инструментом для взаимодействия с systemd. Она используется для запуска, остановки, перезагрузки и мониторинга служб.

Основные команды по управлению сервисами в ОС:

– Запуск сервиса:

```
sudo systemctl start <имя_сервиса>
```

– Остановка сервиса:

```
sudo systemctl stop <имя_сервиса>
```

– Перезапуск сервиса:

```
sudo systemctl restart <имя_сервиса>
```

– Проверка статуса сервиса:

```
systemctl status <имя_сервиса>
```

– Включение автозапуска сервиса при старте Системы:

```
sudo systemctl enable <имя_сервиса>
```

– Отключение автозапуска сервиса при старте Системы:

```
sudo systemctl disable <имя_сервиса>
```

– Перезагрузка с целью применения изменений конфигурации без перезапуска служб:

```
sudo systemctl reload <service_name>
```

– Управление группами служб, например, можно запускать или останавливать группу служб:

```
sudo systemctl start <group_name>
sudo systemctl stop <group_name>
```

Фильтрация вывода systemctl возможна по различным критериям, таким как состояние службы с использованием опции --state или тип с использованием опции –type .

3.4.2 Unit-файлы

Unit-файлы являются основными конфигурационными файлами, которые используются в systemd для управления службами, монтированием файловых систем, сокетами и другими системными объектами. Именно они определяют, как и когда должны запускаться процессы, как обрабатывать их зависимости, и другие аспекты работы службы или ресурса.

Существуют различные типы unit-файлов:

– service (расширение .service) – определяет параметры для управления службами;

– target (расширение .target) – группирует несколько unit-файлов и управляет их совместным запуском;

– socket (расширение .socket) – управляет сокетами, которые используются для активации служб;

– mount (расширение .mount) – управляет точками монтирования файловых систем;

– timer (расширение .timer) – предназначены для создания и управления таймерами, выполняют роль планировщика задач.

Unit-файлы имеют следующую базовую структуру:

– [Unit] – описание метаданных и зависимостей;

– [Service]/ [Timer] (или другой секции в зависимости от типа unit-файла) – определяет параметры запуска и управления процессом;

– [Install] – настройки, связанные с автозапуском и включением unit-файла в зависимости от системных целей (targets).

Настройки параметров unit-файлов помогают детально управлять поведением и запуском различных системных служб и компонентов. Алгоритм написания и настройки unit-файлов в systemd:

а) **Секция [Unit]** задает общие сведения о юните и его зависимости:

– Description= – краткое описание юнита;

– Wants= – указывает на необязательную зависимость: основной юнит пытается запустить другой юнит, указанный в параметре. Ошибки при запуске зависимого юнита не мешают запуску основного. Если параметры After= и Before= не указаны, юниты запускаются параллельно;

– Requires= – обозначает обязательную зависимость: основной юнит требует запуск другого юнита. Если запуск зависимого юнита неудачен, основной юнит также не запустится. Без параметров After= и Before= оба юнита будут запущены одновременно;

– After= – указывает, что основной юнит должен запускаться только после другого, указанного в параметре. Этот параметр часто используется вместе с Wants= и Requires=;

– Before= – противоположен After=, определяет запуск основного юнита перед другим юнитом.

б) **Секция [Install]** используется для настройки автозапуска юнита:

– Alias= – позволяет задать дополнительные имена для юнита, при этом создаются символические ссылки на основной юнит;

Примечание – Поддерживается не всеми типами юнитов.

– WantedBy= – указывает на цель, с которой связан данный юнит: при активации юнита команда `systemctl enable` создаст символическую ссылку в соответствующем каталоге;

– Also= – перечисляет другие юниты, которые также будут добавлены в автозапуск или удалены из него при изменении состояния текущего юнита.

в) **Секция [Service]** задает параметры управления службами:

– Type= – определяет, как служба будет запускаться:

– simple – запускает процесс немедленно, без создания дочерних процессов;

– forking – запускает службу, которая создает дочерний процесс и завершает родительский. Часто используется для демонов;

– oneshot – для одноразовых задач, которые завершаются после выполнения;

– notify – аналогично simple, но с дополнительным уведомлением о готовности;

– dbus – служба считается готовой, когда определенное имя появляется на шине D-Bus;

– idle – откладывает запуск службы до завершения всех других задач;

– PIDFile= – путь к PID-файлу;

– WorkingDirectory= – рабочий каталог, в котором будет работать процесс;

– User= – пользователь, от имени которого запускается служба;

– Group= – группа, от имени которой запускается служба;

– OOMScoreAdjust= – регулирует приоритет процесса при нехватке памяти;

– ExecStop= – команда для остановки службы;

– ExecStart= – команда, которая выполняется при запуске службы;

– RemainAfterExit= – указывает, что служба считается активной даже после завершения.

г) **Секция [Socket]** управляет параметрами сокетов:

– ExecStart= – команда для запуска;

– ExecReload= – команда для перезапуска;

– KillMode= – режим завершения;

– Restart= – правила перезапуска в случае ошибки.

Пример unit-файла службы:

```
[Unit]
Description=Пример службы
After=network.target

[Service]
ExecStart=/usr/bin/example-service
Restart=on-failure

[Install]
```

```
WantedBy=multi-user.target
```

где параметры:

– Description= – предоставляет краткое описание службы;

– After= – указывает, что служба должна быть запущена после достижения определенной цели (например, network.target);

– ExecStart= – определяет команду, которую необходимо выполнить для запуска службы;

– Restart= – управляет политикой перезапуска, в данном случае on-failure означает, что служба перезапустится при её сбое;

– WantedBy= – определяет, в каких режимах работы (targets) Система должна запускать эту службу.

Различные unit-файлы располагаются в следующих каталогах Системы:

– /etc/systemd/system/ или /lib/systemd/system/ – для системных файлов. Файлы в этом каталоге имеют приоритет перед другими;

– /usr/lib/systemd/system/ – для пользовательских unit-файлов;

– /run/systemd/system/ – для временных unit-файлов, которые генерируются во время работы Системы.

3.4.2.1 Планирование задач с файлами .timer

Файлы с расширением .timer предназначены для создания и управления таймерами. В ОС они выполняют роль планировщика задач, являясь частью systemd и заменяя классический cron.

Файлы с расширением .timer работают в паре с .service-файлами, которые описывают действия, которые должны быть выполнены в запланированное время. Таймеры активируют определённые сервисы в моменты, заданные в конфигурации .timer-файла.

Файлы с расширением .timer имеют следующую основную структуру:

```
[Unit]
Description=Описание таймера
Requires=my.service #Запуск таймера, только при наличии соответствующего сервис файла “my.service”

[Timer]
OnCalendar=*-*-* 02:00:00 # Указание расписания (пример: каждый день в 2:00)
Persistent=true # Выполнять задачу после пропуска, если Система была выключена
```

```
    Unit=my.service    #Таймер,  при  срабатывании,  запускает
соответствующий сервис “my.service”

    [Install]
    WantedBy=timers.target      # Активирует файл .target если
есть и запущен timers.target
```

Основные секции файла .timer

а) [Unit] — стандартная секция для всех файлов systemd, содержит описание таймера:

– Description= — краткое описание цели таймера;

– Requires=– запуск таймера произойдет, только если существует соответствующий указанный сервис файл;

б) [Timer] — основная секция, где задаются параметры работы таймера:

– OnCalendar= — задаёт расписание запуска (например, ежедневно, ежемесячно, по конкретным датам);

– OnBootSec= — время после загрузки Системы (например, запуск через 5 минут после загрузки);

– OnUnitActiveSec= — время после последнего запуска службы (например, через 10 минут после последнего выполнения).

– Persistent= — если "true", то задача будет выполнена сразу после включения Системы, если во время её выключения был пропущен запуск.

– Unit= — данный параметр означает, что таймер, при срабатывании, запускает указанный сервис файл.

в) [Install] — содержит информацию о том, как таймер должен быть активирован.

– WantedBy= — если есть и запущен файл timers.target (который активирует все таймеры systemd), то активируется целевой .target файл и начинается отсчет времени таймера.

Элементы расписания OnCalendar

Значения параметра OnCalendar указывают на время или периодичность выполнения задания файла .service.

Формат параметра OnCalendar:

```
    * *-*-* *:*:*
    DayOfWeek Year-Month-Day Hour:Minute:Second
```

Этот формат можно комбинировать или опускать некоторые элементы. Синтаксис параметра поддерживает различные вариации, чтобы можно было точно настроить расписание.

Элементы расписания OnCalendar:

– DayOfWeek — день недели (Sun, Mon, Tue, Wed, Thu, Fri, Sat).

Mon *-*-* 02:00:00 — каждый понедельник в 2:00.

– Year-Month-Day — год, месяц и день.

2024-10-01 — 1 октября 2024 года.

– Использование символов подстановки: * — любая дата (например, "*-*-01" означает первый день каждого месяца).

Запись "*-*-*" означает каждый день в любом году и месяце.

– Hour:Minute — время в формате часов, минут и секунд.

14:30:00 — 14 часов 30 минут ровно.

Если нужно указать только часы и минуты, можно опустить секунды: 14:30.

Параметры опции OnCalendar:

– Ежедневный запуск в полночь:

```
OnCalendar=daily
```

– Еженедельный запуск (по умолчанию воскресенье в 0:00):

```
OnCalendar=weekly
```

– Ежемесячный запуск (в полночь первого числа каждого месяца):

```
OnCalendar=monthly
```

– Ежегодный запуск (первого января каждого года).

```
OnCalendar=yearly
```

или

```
OnCalendar=annually
```

– Запуск каждый день в 2:00 ночи:

```
OnCalendar=*-*-* 02:00:00
```

– Запуск по будням в 8:00 утра:

```
OnCalendar=Mon..Fri *-*-* 08:00:00
```

– Запуск в первый день каждого месяца ровно в полночь:

```
OnCalendar=*-*-01 00:00:00
```

– Запуск каждый год в конкретный день (например, 25 декабря каждого года):

```
OnCalendar=*-12-25 00:00:00
```

– Запуск раз в час:

```
OnCalendar=hourly
```

– Запуск каждые 15 минут:

```
OnCalendar=*:00/15
```

Если таймер содержит опцию Persistent=true, это означает, что, если Система была выключена в момент времени, когда задача должна была запуститься, она будет выполнена сразу после следующего включения. Это полезно для гарантий выполнения важных задач.

systemd позволяет комбинировать параметры настроек. Например:

```
OnCalendar=Mon *-*-* 03,15:00:00
```

Здесь задается запуск задачи каждый понедельник в 3:00 и 15:00.

Чтобы проверить правильность работы таймера, можно использовать команду:

```
systemctl list-timers
```

Она покажет все таймеры, когда они были последовательно выполнены и когда будут выполнены в следующий раз.

3.4.2.2 Работа таймеров с файлами .service

Для работы таймеров необходимо создать соответствующий файл WantedBy=multi-user.target.service, который описывает, что должно выполняться, когда таймер срабатывает.

Файл .timer определяет расписание запуска.

Файл .service определяет, что нужно выполнить по этому расписанию.

Таймер запускает указанный сервис в момент времени, заданный в .timer файле.

Для примера рассмотрим файл .service для выполнения задачи резервного копирования:

```
[Unit]
Description=Задача резервного копирования
Wants=my.timer

[Service]
Type=oneshot
ExecStart=/usr/bin/rsync -av /home/ /mnt/backup
```

```
[Install]
WantedBy=multi-user.target
```

где:

– Параметр Wants=my.timer указывает, что необходимый сервис запустится и будет в режиме ожидания если запущен и работает таймер my.timer.

– Параметр WantedBy=multi-user.target указывает, на каком уровне загрузки Системы будет работать данный сервис.

Основные команда для управления таймерами:

– Запуск таймера вручную:

```
sudo systemctl start backup.timer
```

– Активация таймера при запуске Системы:

```
sudo systemctl enable backup.timer
```

– Проверка статуса таймера:

```
systemctl status backup.timer
```

– Просмотр всех активных таймеров:

```
systemctl list-timers
```

3.4.3 Состояние Системы

Состояние Системы и набор загружаемых сервисов определяется целями (targets) systemd. Цели (targets) systemd представляют собой группы служб и юнитов, которые должны быть активированы в определенных состояниях Системы.

Основные цели systemd:

– poweroff.target – используется для полного отключения Системы.

– rescue.target – вводит Систему в однопользовательский режим для диагностики и восстановления. Этот режим предназначен для выполнения административных задач, доступных только для root.

– multi-user.target – полный многопользовательский режим с поддержкой сети. Эта цель запускает текстовый интерфейс, и большинство серверов работают именно в этом режиме.

– graphical.target – запускает Систему в многопользовательском режиме с графическим интерфейсом (GUI). Этот режим используется на настольных системах.

– reboot.target – перезагружает Систему.

Помимо стандартных целей, systemd включает и другие специализированные цели:

– emergency.target – самый минималистичный режим с доступом только к корневой консоли. Используется для восстановления в критических ситуациях.

– halt.target – останавливает Систему без полного выключения (зависает на экране "halt").

– suspend.target, hibernate.target, hybrid-sleep.target – Эти цели управляют различными состояниями энергосбережения, такими как приостановка (suspend), гибернация (hibernate) и гибридный сон.

Цели (targets) в systemd определены в виде файлов с расширением .target. Системные файлы .target расположены в каталоге /lib/systemd/system/ или /etc/systemd/system/ и отвечают за запуск Системы в указанном состоянии. Пользовательские файлы .target хранятся в /usr/lib/systemd/user/.

Переключаться между целями можно с помощью команды:

```
sudo systemctl isolate <цель>
```

Например, чтобы перевести Систему в графический режим можно использовать команду:

```
sudo ystemctl isolate graphical.target
```

Чтобы просмотреть активную цель можно использовать команду:

```
sudo systemctl get-default
```

3.4.4 systemd-resolved

systemd-resolved — системная служба, предназначенная для разрешения сетевых имён, входящая в состав пакета systemd. Служба обеспечивает поддержку различных протоколов разрешения имён, включая:

– традиционную систему доменных имён (DNS), в том числе DNSSEC и DNS over TLS;

– Multicast DNS (mDNS);

– Link-Local Multicast Name Resolution (LLMNR).

3.4.4.1 Поддерживаемые режимы работы

Служба systemd-resolved может функционировать в одном из четырёх режимов. Наиболее часто применяются следующие два:

– Работа через DNS-заглушку systemd (рекомендуемый режим): в этом режиме используется локальный адрес 127.0.0.53 в качестве единственного DNS-сервера. Необходимо создать символическую ссылку:

```
ln -sf /run/systemd/resolve/stub-resolv.conf\
/etc/resolv.conf
```

Это обеспечивает корректную работу всех процессов с использованием systemd-resolved.

– Работа с прямым использованием файла /etc/resolv.conf: в этом режиме служба использует указанный файл напрямую, как обычный DNS-клиент. Этот режим подходит для ситуаций, когда требуется сохранить совместимость с устаревшими приложениями.

Режим работы определяется автоматически, в зависимости от содержимого или назначения символической ссылки /etc/resolv.conf.

Для получения сведений о текущих настройках systemd-resolved, включая активные DNS-серверы, используется команда:

```
resolvectl status
```

Служба работает автоматически с сетевыми менеджерами, использующими файл /etc/resolv.conf, в том числе с systemd-networkd и NetworkManager. При этом DNS-серверы передаются службе systemd-resolved автоматически.

При необходимости можно вручную задать параметры DNS-серверов и fallback-серверов в файле /etc/systemd/resolved.conf. Пример настройки:

```
[Resolve]
DNS=192.168.35.1 fd7b:d0bd:7a6e::1
FallbackDNS=208.67.222.222 208.67.220.220
```

В случае отсутствия информации от сетевого менеджера и при неуказанных вручную серверах служба использует зарезервированные DNS-адреса (например, "Яндекс 77.88.8.8", "Google 8.8.8.8").

Управление службой systemd-resolved:

– перезапуск службы:

```
sudo systemctl restart systemd-resolved.service
```

– остановка службы:

```
sudo systemctl stop systemd-resolved.service
```

По умолчанию служба systemd-resolved включена и активна.

3.4.4.2 Конфликт порта 53 при использовании systemd-resolved

Служба systemd-resolved по умолчанию использует порт 53 (протокол DNS) на локальном интерфейсе 127.0.0.53 для организации локальной DNS-заглушки. В случае установки и запуска на том же узле стороннего DNS-сервера (например, named в составе BIND), также предназначенного для работы на порту 53, возникает конфликт ресурсов: обе службы не могут одновременно прослушивать один и тот же порт.

Подобная ситуация характерна, в частности, при настройке контроллера домена Samba с использованием внешнего DNS-сервера. Для предотвращения конфликта требуется отключить использование порта 53 службой systemd-resolved.

Для отключения порта 53 необходимо выполнить следующие действия:

а) Создать каталог для дополнительной конфигурации службы:

```
mkdir -p /etc/systemd/resolved.conf.d
```

б) Создать в нём файл с настройкой, отключающей локальную DNS-заглушку:

```
echo -e '[Resolve]\nDNSStubListener=no' >\
/etc/systemd/resolved.conf.d/no53port.conf
```

в) Перезапустить службу systemd-resolved, чтобы применить изменения:

```
sudo systemctl try-restart systemd-resolved
```

После выполнения указанных действий служба systemd-resolved прекращает использовать порт 53, что позволяет другим компонентам (например, named) корректно его задействовать.

3.5 Файловые системы

Файловая система (ФС) определяет способ организации и хранения данных на носителях информации, таких как жесткие диски, SSD и USB. Она управляет тем, как данные записываются, хранятся и читаются с устройств хранения, предоставляя структуру каталогов и файлов.

ОС поддерживает различные файловые системы (ФС) для обеспечения гибкости и оптимальной производительности при работе с дисками и разделами. По умолчанию используется журналируемая файловая система Ext4. Однако во время установки Системы администратору предоставляется возможность выбора других файловых систем (см. раздел 2.2.2), таких как Ext3, XFS, и BTRFS. Также поддерживается работа с дисками, отформатированными в NTFS (файловая система Windows).

3.5.1 Поддерживаемые файловые системы

Рассмотрим общие аспекты поддерживаемых в Системе ФС:

– **Ext4** (Fourth Extended Filesystem) является основной и рекомендованной ФС для ОС. Она представляет собой усовершенствованную версию Ext3, предлагая улучшенную производительность и дополнительные функции, такие как журналирование, поддержка больших файлов и разделов, быстрое выделение места для файлов и защита от фрагментации. Данная ФС предпочтительна при работе в ОС поскольку является стабильным и широко поддерживаемым выбором для большинства пользователей.

Журналирование обеспечивает сохранность данных в случае сбоев. Все изменения в каталогах сначала записываются в журнал и только после этого применяются к основным данным. Это гарантирует целостность файловой системы и минимизирует риск повреждения данных

Ext4 включает механизм автоматического выравнивания блоков данных и индексных записей, что помогает поддерживать высокую производительность даже в случае фрагментированных данных.

Как и в других UNIX-подобных системах, в Ext4 доступ к каталогам управляется через систему прав доступа (rwx).

– **Ext3** (Third Extended Filesystem) является предшественником Ext4 и также поддерживается ОС РОСА "ХРОМ". Ее основное преимущество — журналирование, но она менее производительна и обладает меньшими возможностями по сравнению с Ext4. Ext3 может использоваться для совместимости с более старыми системами или программами.

– **XFS** — это высокопроизводительная файловая система, разработанная компанией Silicon Graphics. Она отличается хорошей масштабируемостью и эффективной работой с большими файлами и разделами. XFS также обладает функциями журналирования, быстрым выделением пространства для файлов и достаточно высокой производительностью при записи данных. Однако она не поддерживает нативное создание снимков (snapshot), что ограничивает ее функциональность в современных сценариях резервного копирования и восстановления данных.

– **NTFS** (New Technology File System) — файловая система, разработанная Microsoft для ОС Windows. В ОС РОСА "ХРОМ" поддержка NTFS реализована через встроенные драйверы ядра, которые позволяют монтировать и работать с NTFS-разделами напрямую. NTFS часто используется для обмена данными между Linux-подобными ОС и ОС семейства Windows.

3.5.1.1 Просмотр поддерживаемых файловых систем

Для того чтобы просмотреть список всех ФС, поддерживаемых в ОС, нужно выполнить команду:

```
cat /proc/filesystems
```

В результате выполнения данной команды будет выведен список всех поддерживаемых ФС (рисунок \# 013).

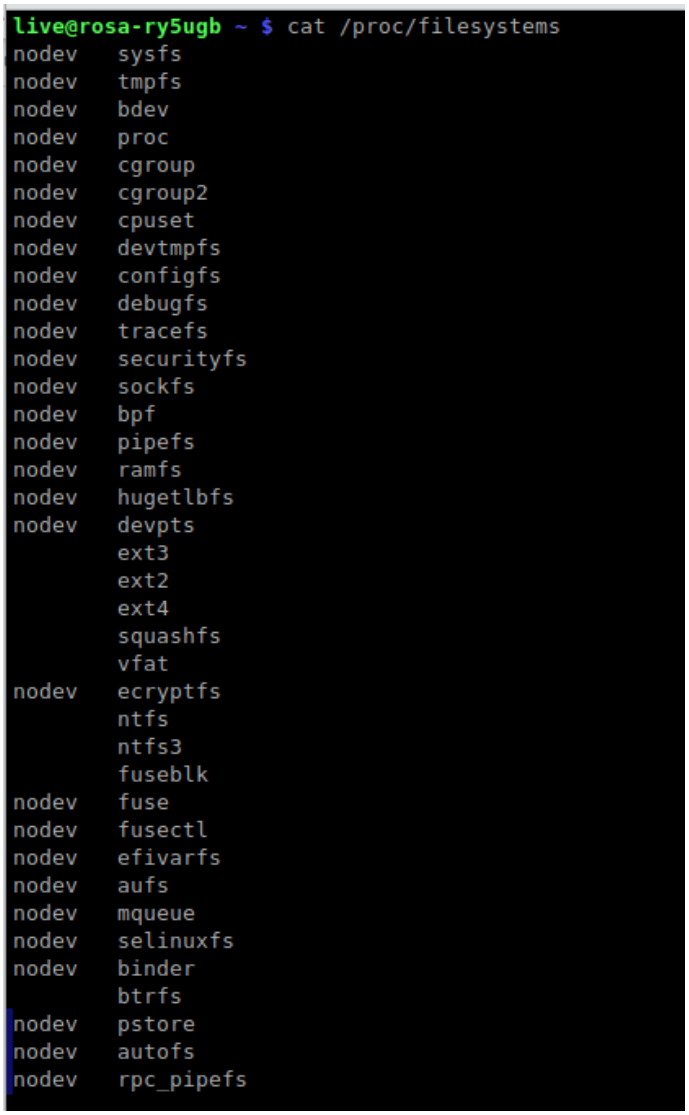


Рисунок 13 — Список поддерживаемых ФС

Использование команды `cat /proc/filesystems` позволяет определить все ФС, которые поддерживаются в Системе.

Для поиска конкретной ФС можно использовать следующую команду (рисунок 14):

```
cat /proc/filesystems | grep btrfs
```

где btrfs – название искомой ФС.

```
live@rosa-ry5ugb ~ $ cat /proc/filesystems | grep btrfs
        btrfs
```

Рисунок 14 — Поиск BTRFS в списке поддерживаемых ФС

3.5.2 Организация каталогов в Системе

Каталоги в файловой системе Ext4 являются специальным типом файлов, которые содержат список имен файлов и их соответствующих inode-номеров. В inode хранится информация о файле или каталоге, включая права доступа, владельца, размер, а также указатели на блоки данных.

В Ext4 каталоги организованы как древовидная структура, где каждый каталог может содержать файлы и подкаталоги. Корневой каталог / является начальной точкой этой структуры. Внутри него находятся системные каталоги, такие как /bin, /etc, /home, /var и другие.

В Ext4 реализована схема HTree (сбалансированное дерево), которая позволяет увеличить скорость поиска и доступа к файлам в больших каталогах.

Пределы и ограничения на хранение данных в каталогах ФС:

– Ext4 поддерживает каталоги размером до 2 ТБ, что существенно увеличивает их емкость по сравнению с Ext3.

– В одном каталоге Ext4 может содержать до 10^{18} файлов, что практически не ограничивает количество файлов в каталоге.

– Стандартная структура каталогов в ОС позволяет администраторам и пользователям эффективно управлять Системой и поддерживать её в рабочем состоянии.

Основные каталоги ОС, которые используются Системой по умолчанию:

– / (Корневой каталог) – начальная точка всей ФС, все другие каталоги и файлы находятся внутри этого каталога;

– /bin – содержит основные исполняемые файлы (команды), необходимые для работы Системы, которые могут использовать все пользователи (Например: bash, ls, cp);

– /boot – содержит файлы, необходимые для загрузки системы, такие как ядро, загрузчик и начальный RAM-диск;

– /dev – содержит файловые интерфейсы к устройствам (драйверам). Здесь находятся файлы устройств, которые представляют периферийные устройства, такие как жесткие диски, терминалы и другие;

– /etc – содержит конфигурационные файлы Системы и служб. Это место, где хранятся конфигурации практически всех программ и служб на Системе;

– /home – содержит домашние каталоги пользователей. Каждый пользователь имеет свой собственный подкаталог, например /home/user;

– /lib – содержит общие библиотеки, используемые системными программами, и модули ядра;

– /media – место для автоматического монтирования съемных носителей, таких как CD-ROM, USB и другие;

– /mnt – каталог, используемый для временного монтирования файловых систем;

– /opt – каталог для установки дополнительного ПО, которое не входит в стандартный набор программ ОС;

– /proc – виртуальная файловая система, предоставляющая информацию о текущем состоянии Системы, включая информацию о процессах;

– /root – домашний каталог пользователя root (суперпользователя);

– /run – каталог для временных данных, которые нужны при запуске Системы, таких как PID-файлы;

– /sbin – содержит системные программы, которые обычно используются только администратором Системы (root);

– /srv – каталог, предназначенный для хранения данных, которые обслуживаются Системой, например, веб-сервером или FTP-сервером;

– /tmp – каталог для временных файлов. Эти файлы обычно удаляются при перезагрузке Системы;

– /usr – содержит вторичные программы, библиотеки и документацию. Внутри него также находятся такие подкаталоги, как:

– /usr/bin – дополнительные исполняемые файлы;

– /usr/lib – дополнительные библиотеки;

– /usr/share – общие файлы данных, такие как значки и документация;

– /usr/local – место для установки программного обеспечения вручную, которое не будет затронуто при обновлении Системы;

– /var – каталог, содержащий данные, которые изменяются во время работы Системы, такие как логи, файлы временного хранения, очереди задач и т.д.

В пользовательских каталогах наряду с персональными файлами также имеются и конфигурационные файлы, которые в целях безопасности скрыты. Файлы и каталоги считаются скрытым, если их имена начинаются с точки (.). (например, файл .config является скрытым).

Для того чтобы увидеть скрытые файлы в командной строке, можно использовать команду `ls` с параметром -a:

```
ls -a
```

Чтобы сделать файл или каталог скрытым, достаточно переименовать его, добавив в начало имени точку. Например, можно переименовать файл filename в .filename:

```
mv filename .filename
```

3.5.3 Монтирование

Прежде чем приступить к работе с ФС, её нужно смонтировать, что означает подключение к файловой структуре Системы. Этот процесс позволяет ОС правильно взаимодействовать с ФС, делая её доступной как обычный каталог, известный как точка монтирования.

Для успешного монтирования ФС, важно убедиться, что каталог, выбранный в качестве точки монтирования, существует в Системе. Если точка монтирования является не пустым каталогом, его содержимое будет недоступно до тех пор, пока файловая система не будет размонтирована. Поэтому рекомендуется использовать специально созданные каталоги для этой цели, которые обычно находятся в каталоге /mnt или /media.

Допустим, необходимо смонтировать ISO-образ в каталог /mnt. Этот каталог должен существовать монтирования, иначе операция завершится неудачно. После успешного монтирования в каталоге появится содержимое ФС, в противном случае каталог останется пустым.

Чтобы определить, к какой ФС относится текущий каталог и объём свободного места на ней, используют команду:

```
df -h
```

В ОС для монтирования ФС используется команда `mount`. Общий синтаксис команды:

```
sudo mount устройство точка_монтирования
```

где:

– устройство — это физический носитель данных, который необходимо подключить;

– точка_монтирования — каталог, к которому будет привязана ФС.

Из соображений безопасности команда `mount` доступна для выполнения только с правами суперпользователя. Команда `mount` может включать дополнительные параметры, такие как:

– -f – имитирует монтирование, выполняя все действия, кроме фактического монтирования файловой системы;

– -v – предоставляет подробный отчёт о выполняемых действиях;

– -w – монтирует файловую систему с доступом для чтения и записи;

– -r – монтирует файловую систему только для чтения;

– -n – монтирует без записи в файл /etc/mtab;

– -t type – указывает тип монтируемой файловой системы;

– -a – монтирует все файловые системы, указанные в /etc/fstab;

– -o list_of_options – применяет список опций к файловой системе, где опции разделены запятыми, подробнее об этом см. в man mount.

Подробное описание опций команды `mount` доступно в руководстве man mount.

Если у команды `mount` не указаны опции, то она попытается определить их по файлу /etc/fstab.

Примеры команд монтирования:

– Монтирование раздела жесткого диска:

```
sudo mount /dev/hdb3 /mnt
```

– Монтирование всех ФС NFS, указанных в /etc/fstab:

```
sudo mount -vat nfs
```

Если возникли ошибки при монтировании, можно использовать команду:

```
sudo mount -vf устройство точка_монтирования
```

Эта команда выполнит все действия, кроме монтирования, и выдаст подробный отчет о каждом шаге.

Файл /etc/fstab — это конфигурационный файл, который определяет порядок монтирования файловых систем в ОС. Этот файл играет ключевую роль в автоматическом подключении устройств и разделов при загрузке Системы, а также упрощает управление монтированием для системных администраторов.

Файл /etc/fstab состоит из строк, каждая из которых описывает отдельную файловую систему или устройство. Рассмотрим типовую структуру файла fstab:

```
    # # <файловая система>     <точка монтирования>     <тип>
<параметры>   <dump>   <pass>
    UUID=abcd1234-ef56-7890-abcd-1234567890ab   /           ext4
defaults        1 1
    UUID=abcd1234-ef56-7890-abcd-1234567890ac   /home       ext4
defaults        1 2
    UUID=abcd1234-ef56-7890-abcd-1234567890ad   /var         xfs
defaults        1 2
    UUID=abcd1234-ef56-7890-abcd-1234567890ae   swap        swap
defaults        0 0
```

```
/dev/cdrom      /media/cdrom  auto    noauto,ro     0 0
```

Каждая строка включает следующие поля:

– <файловая система> — идентификатор файловой системы, которая должна быть смонтирована. Обычно это UUID (уникальный идентификатор) раздела, но может быть также использован путь к блочному устройству (например, /dev/sda1).

В рассмотренном выше примере строка: UUID=abcd1234-ef56-7890-abcd-1234567890ab, /dev/cdrom.

– <точка монирования> — каталог, в который будет подключена файловая система.

В рассмотренном выше примере: /, /home, /var, /media/cdrom.

– <тип> — тип файловой системы, который будет использоваться при монтировании. Это может быть ext4, xfs, btrfs, swap для разделов подкачки, или auto для автоматического определения типа.

В рассмотренном выше примере: ext4, xfs, swap, auto.

– <параметры> — монтировочные опции, которые определяют поведение файловой системы. Наиболее часто используются опции:

– defaults — включает стандартный набор опций: rw, suid, dev, exec, auto, nouser, async.

– noauto — файловая система не монтируется автоматически при загрузке.

– ro — монтирование в режиме только для чтения.

В рассмотренном выше примере: defaults, noauto,ro.

– <dump> — опция для утилиты dump, которая выполняет резервное копирование файловой системы. Если установлено в значение "1", резервное копирование включено, если в "0" — отключено.

В рассмотренном выше примере: 1, 0.

– <pass> — определяет порядок проверки файловых систем при загрузке:

– 0 — файловая система не будет проверяться.
– 1 — корневая файловая система проверяется первой.
– 2 — другие файловые системы проверяются после корневой.

В рассмотренном выше примере: "1", "2", "0".

Редактирование файла /etc/fstab требует прав суперпользователя. Изменения в этом файле немедленно не применяются, поэтому после редактирования необходимо либо вручную смонтировать файловые системы командой `sudo mount -a`, либо перезагрузить Систему для применения настроек.

Чтобы избежать ошибок при редактировании /etc/fstab, рекомендуется тщательно проверять правильность введенных данных, так как ошибки могут привести к тому, что Система не сможет загрузиться корректно.

Для получения полной информации о возможных опциях для файла /etc/fstab смотрите справочную страницу по команде `man fstab`.

3.5.4 Размонтирование

Для отключения файловой системы от ОС используется команда `umount`. Операция размонтирования может потребоваться для выполнения проверок и восстановления ФС с использованием команды `fsck`.

В случае проблем с сетью, удаленные ФС также могут быть отключены.

Команда `umount` имеет следующий формат:

```
umount device : mountpoint
umount -a
umount -t fstype
```

Здесь device обозначает физическое устройство, которое нужно отключить, а mountpoint — каталог, к которому оно подключено (можно указать только device или mountpoint). Параметр -a отключает все подключенные ФС, а параметр -t fstype отключает ФС указанного типа.

Команда `umount` не сможет отключить ФС, если она в данный момент используется. Например, если ФС подключена к /mnt, и вы можете попытаться выполнить следующие команды:

```
cd /mnt
sudo umount /mnt
```

В результате будет получено сообщение об ошибке, так как Система будет считать, что ФС занята. Чтобы успешно отключить /mnt, необходимо сначала выйти из этого каталога, перейдя в другой каталог.

Для принудительного отключения устройства, даже если оно используется, можно применить опцию -f:

```
sudo umount -f /cdrom
```

Для безопасного извлечения сменных носителей можно использовать команду `eject`.

Более подробное описание команды доступно в справочном руководстве по команде:

```
man umount
```

3.5.5 Файловая система BTRFS

Файловая система BTRFS (B-Tree File System) — современная копируемая при записи (COW) файловая система, разработанная для повышения надежности данных и управления их объемом. Несмотря на то, что она поддерживает все аналогичные функции, как и Ext4, в некоторых сценариях (особенно на серверах) производительность BTRFS может уступать Ext4, особенно при высокой нагрузке на запись или в случае большого количества мелких файлов, а также при создании множества снимков. Документация и поддержка BTRFS могут быть менее обширными и детализированными, чем у Ext4, что может вызывать затруднения при администрировании ОС.

ОС включает поддержку BTRFS в составе ядра и репозитория пакетов, что позволяет использовать данную ФС без необходимости сборки модулей вручную.

Основные особенности эксплуатации BTRFS:

– При активном использовании снимков на HDD возможна сильная фрагментация. Фрагментация на copy-on-write файловых системах, к которым относится и BTRFS, неизбежна и приводит к замедлению работы. На SSD от фрагментации нет таких негативных эффектов.

– При активном использовании снимков в сочетании с низким количеством IOPS ядро ОС может не успевать в фоне сделать то, что делает btrfs balance при запуске вручную. Из-за этого может показаться, что свободного места достаточно, а при попытке записи его не будет хватать. Эту проблему можно решить балансировкой — перераспределением места между данными и метаданными.

– Существует набор скриптов и таймеров systemd btrfsbalance в одноименном пакете (доступен в репозитории) для регулярного выполнения балансировки.

– В репозитории доступны утилиты just-backup-btrfs и btrfs-sync, которые можно использовать для автоматизации создания снимков и при необходимости их передачи на удаленный сервер (передается дельта изменений, а не весь снимок целиком).

– BTRFS проверяет целостность блоков данных при их чтении, поэтому в случае их повреждения будет выведена ошибка.

– При установке ОС на BTRFS через программу установки Anaconda отдельный раздел /boot необязателен. Можно сделать подтом root с точкой монтирования / для удобства создания снимков в дальнейшем (так выполняется автоматическая разметка в BTRFS).

Полный список всех доступных команд и основных опций для работы с файловой системой BTRFS можно посмотреть с помощью команды `btrfs help` (рисунок 15).

```
live@rosa-ry5ugb ~ $ btrfs help
usage: btrfs [global] <group> [<group>...] <command> [<args>]

Global options:
  --format <format> if supported, print subcommand output in that format (text, json)
  -v|--verbose      increase verbosity of the subcommand
  -q|--quiet        print only errors
  --log <level>     set log level (default, info, verbose, debug, quiet)
  --dry-run         if supported, do not do any active/changing actions

Options for the main command only:
  --help            print condensed help for all subcommands
  --version         print version string

    btrfs balance start [options] <path>
        Balance chunks across the devices
    btrfs balance pause <path>
        Pause running balance
    btrfs balance cancel <path>
        Cancel running or paused balance
    btrfs balance resume <path>
        Resume interrupted balance
    btrfs balance status [-v] <path>
        Show status of running or paused balance
```

Рисунок 15 — Список всех доступных команд и опций

3.5.5.1 Создание ФС

Создание файловой системы BTRFS выполняется командой:

```
sudo mkfs.btrfs /dev/sdb
```

где:

– mkfs.btrfs — утилита для создания файловой системы BTRFS;
– /dev/sdb — диск, на котором нужно создать ФС.

В результате выполнения команды `sudo mkfs.btrfs /dev/sdb` может появиться ошибка (рисунок 16). Она возникает потому, что Система не может определить существующую ФС на диске /dev/sdc или на диске есть повреждённая таблица разделов.

```
live@rosa-ry5ugb ~ $ sudo mkfs.btrfs /dev/sdc
btrfs-progs v6.9.2
See https://btrfs.readthedocs.io for more information.

ERROR: probe of /dev/sdc failed, cannot detect existing filesystem
WARNING: cannot read superblock on /dev/sdc, please check manually

ERROR: use the -f option to force overwrite of /dev/sdc
```

Рисунок 16 — Ошибка создания файловой системы

Для исправления ошибки нужно проверить наличие диска /dev/sdb в Системе с помощью команды `sudo lsblk` (рисунок 17).

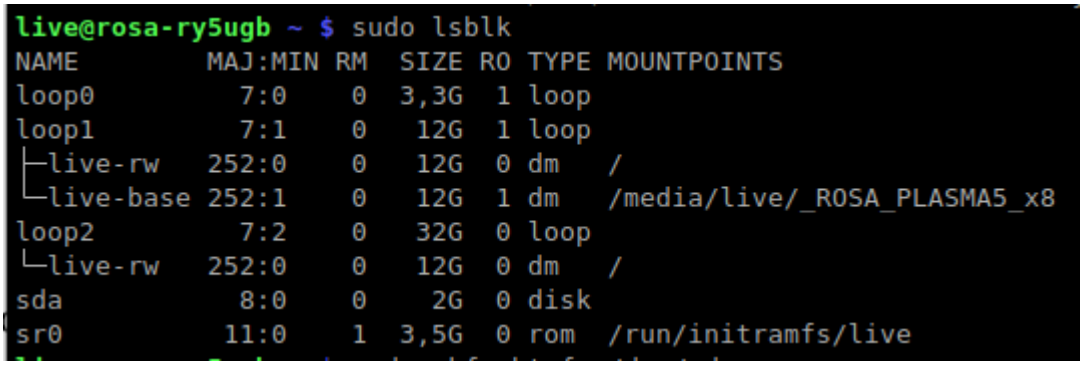


Рисунок 17 — Список существующих дисков и разделов

В результате выполнения команды `sudo lsblk` видно, что диска /dev/sdb не существует. Вместо него используется диск /dev/sda, на котором и нужно создать файловую систему с помощью команды `sudo mkfs.btrfs /dev/sda` (рисунок 18).

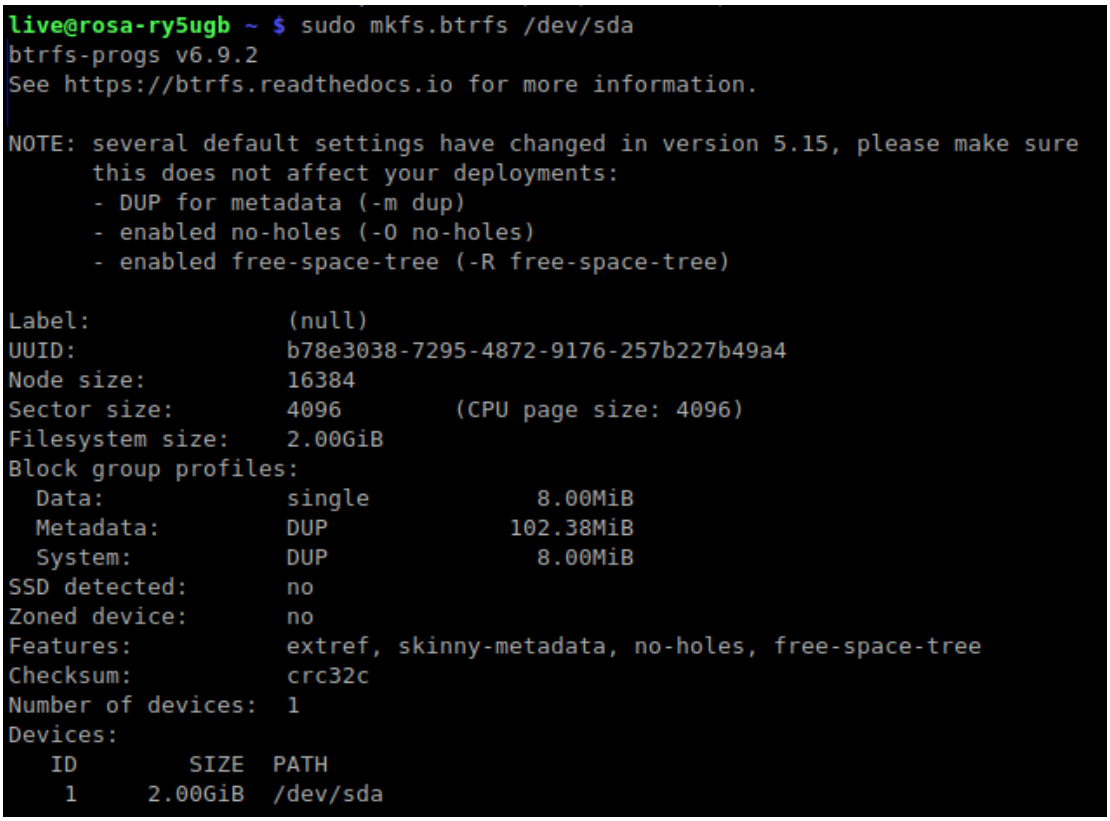


Рисунок 18 — Успешное выполнение команды создания файловой системы

В результате выполнения команды `sudo mkfs.btrfs /dev/sda` файловая система BTRFS будет успешно создана на диске /dev/sda.

Важно — Если диск /dev/sdb существует, но команда `sudo mkfs.btrfs /dev/sdb` не создает ФС, значит на нем повреждена таблица разделов. В этом случае можно выполнить команду `sudo mkfs.btrfs -f /dev/sdc`, в которой используется ключ "-f". С помощью этого ключа BTRFS будет создана, но существующие данные будут удалены с диска.

Для проверки версии BTRFS используется команда (рисунок 19):

```
btrfs --version
```

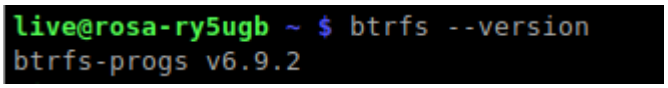


Рисунок 19 — Проверка версии BTRFS

Для определения UUID и типа файловой системы на разделе используется следующая команда (рисунок 20):

```
sudo blkid /dev/sda
```

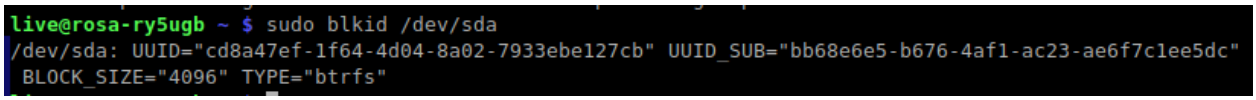


Рисунок 20 — Проверка типа BTRFS

3.5.5.1 Проверка целостности ФС

Для проверки целостности файловой системы можно выполнить одно из следующих действий:

– использовать команду `sudo btrfs check /dev/sda`, если монтирование не проводилось (рисунок 21);

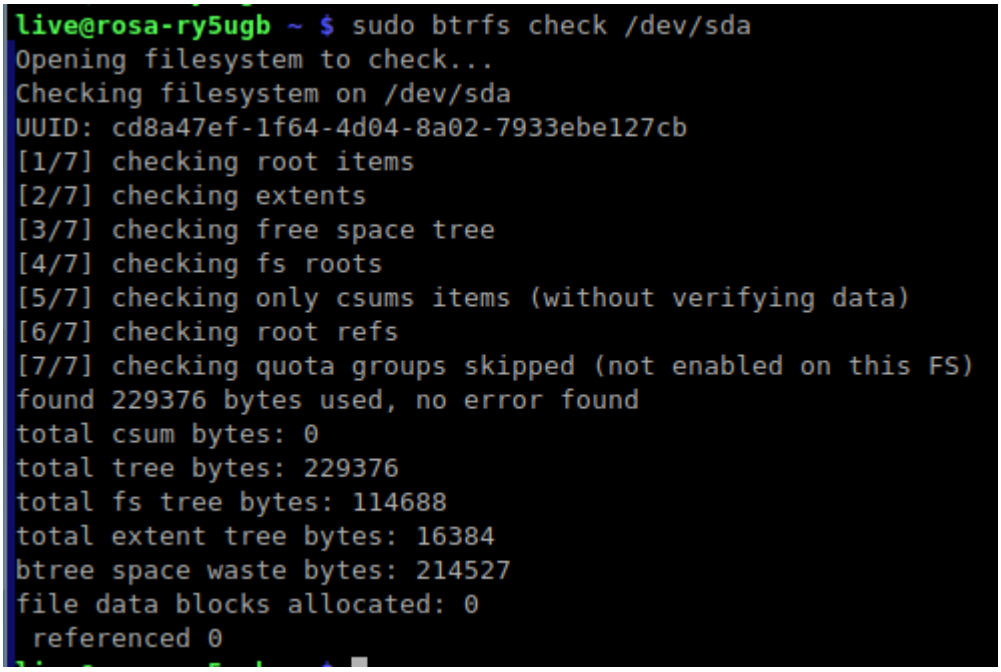


Рисунок 21 — Проверка целостности ФС, если монтирование не проводилось

– использовать команды `sudo btrfs scrub start /mnt/btrfs_disk` и `sudo btrfs scrub status /mnt/btrfs_disk`, если монтирование проводилось (рисунок 22).

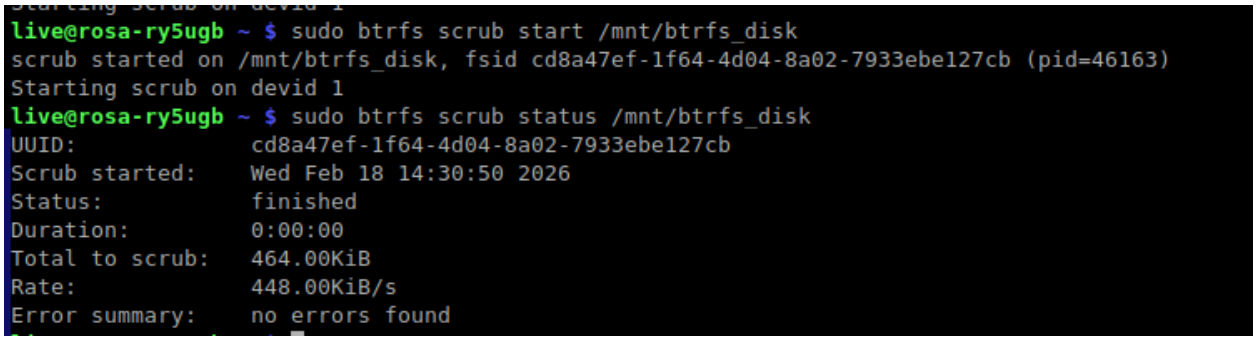


Рисунок 22 — Проверка целостности ФС, если монтирование проводилось

Если в результате выполнения команд появляется строка "no error found", значит проверка прошла успешно.

3.5.5.2 Ручное монтирование

Ручное монтирование подходит для быстрых проверок, но не для долгосрочной работы, поскольку результат не сохраняется после перегрузки Системы.

Ручное монтирование можно выполнить с помощью следующей команды:

```
sudo mount /dev/sda /mnt
```

где:

– /mnt — папка, в которую будет примонтирован диск /dev/sda.

Проверить успешность монтирования можно командой `lsblk` (рисунок 23).

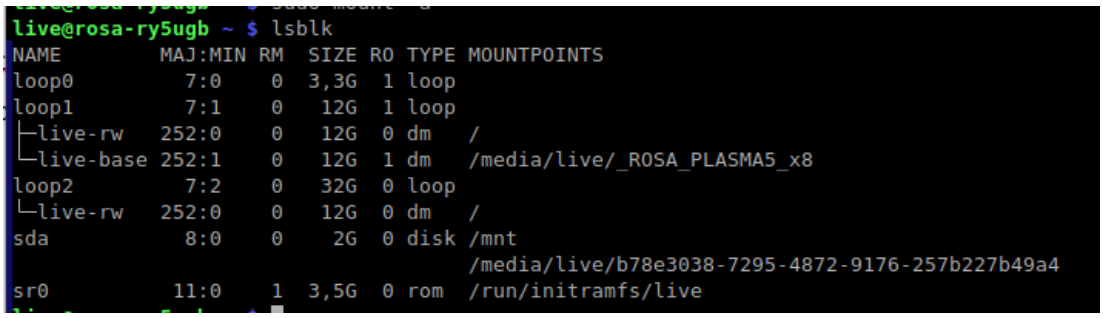


Рисунок 23 — Список существующих дисков и разделов после монтирования

В результате выполнения команды `lsblk` видно, что диск /dev/sda был успешно примонтирован к папке /mnt.

3.5.5.3 Автоматическое монтирование

Для долгосрочной работы рекомендуется настроить автоматическое монтирование, при котором диск привязывается к точке монтирования через UUID.

Чтобы настроить автоматическое монтирование нужно выполнить следующие действия:

а) создать точку монтирования с помощью команды:

```
sudo mkdir -p /mnt/btrfs_disk
```

где:

– -p – флаг, необходимый для создания всех недостающих папок в пути, если бы их не было;

б) получить UUID диска с помощью команды (рисунок 24):

```
sudo blkid /dev/sda
```

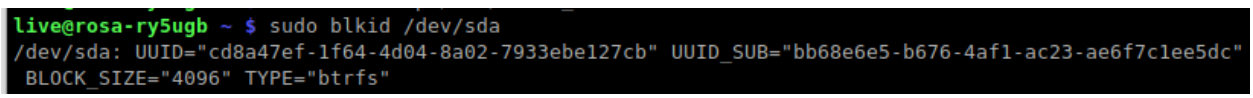


Рисунок 24 — Получение UUID диска

Примечание — Для надежного монтирования не следует использовать имя диска /dev/sda, потому что оно может измениться, а UUID диска остаётся неизменным.

в) отредактировать файл /etc/fstab, в котором хранятся правила автоматического монтирования дисков при загрузке Системы. Для этого нужно использовать команду `sudo nano /etc/fstab`, в результате выполнения которой файл /etc/fstab откроется в редакторе nano (рисунок 25);

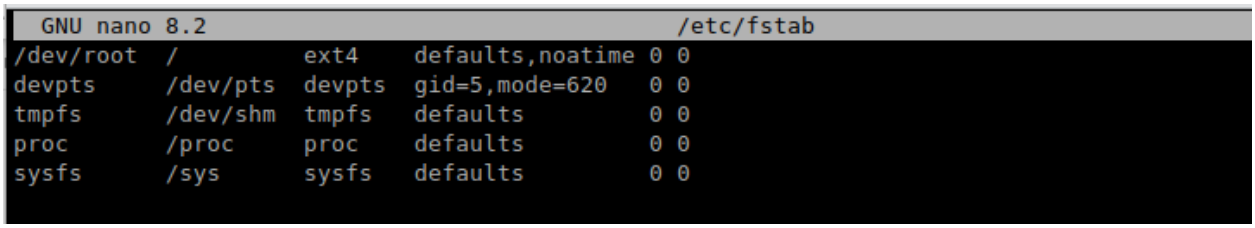


Рисунок 25 — Файл /etc/fstab в редакторе nano

г) вписать строку "UUID=cd8a47ef-1f64-4d04-8a02-7933ebe127cb /mnt/btrfs_disk btrfs defaults 0 0" в конце файла (рисунок 26);

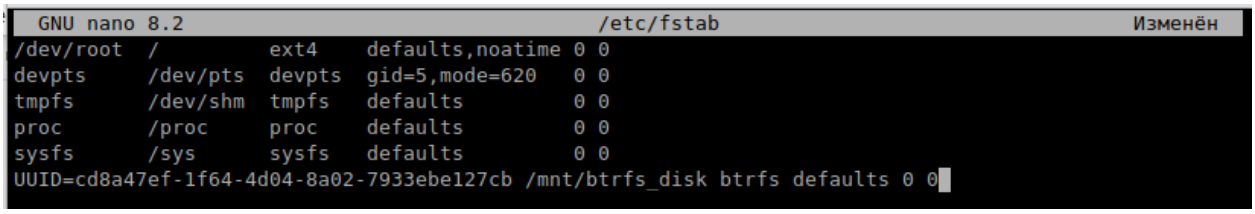


Рисунок 26 — Редактирование файла /etc/fstab

д) использовать сочетание клавиш `Ctrl+S` для сохранения изменений;

е) выйти из редактора nano с помощью сочетания клавиш `Ctrl+X`;

ж) выполнить монтирование всех разделов с помощью команды `sudo mount -`;

з) проверить успешность монтирования с помощью команды `mount | grep btrfs` (рисунок 27);

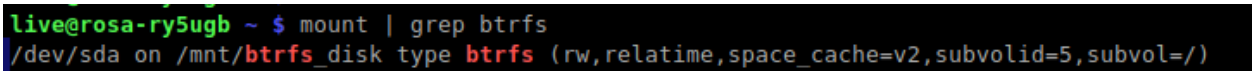


Рисунок 27 — Успешно выполненное монтирование

3.5.5.1 Балансировка

Балансировка в BTRFS — это процесс перераспределения данных и метаданных по блочным устройствам. Она необходима для оптимизации использования пространства и поддержания производительности.

Для проверки работы балансировки необходимо:

а) создать демонстрационную нагрузку с помощью команды (рисунок 28):

```
sudo dd if=/dev/zero of=/mnt/btrfs_disk/@/largefile bs=1M count=100
```

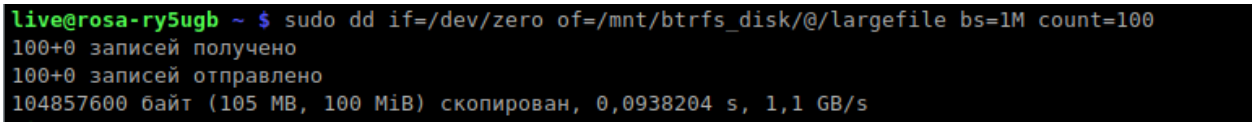


Рисунок 28 — Создание нагрузки

б) просмотреть статистику использования с помощью команды (рисунок 29):

```
sudo btrfs filesystem show /mnt/btrfs_disk
```

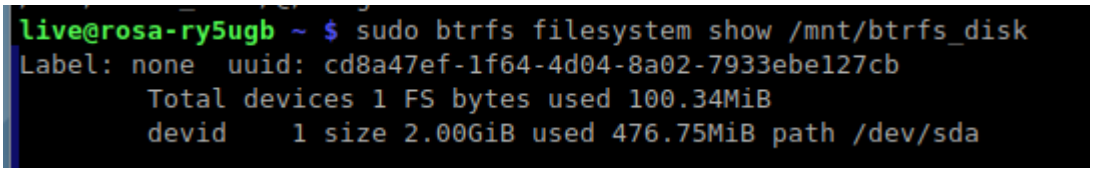


Рисунок 29 — Просмотр статистики использования

в) просмотреть распределение данных с помощью команды (рисунок 30):

```
sudo btrfs filesystem df /mnt/btrfs_disk
```

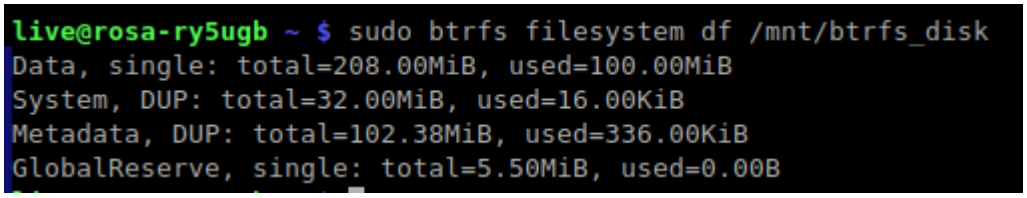


Рисунок 30 — Просмотр распределения данных

г) запустить балансировку с ограничением по заполнению командой:

```
sudo btrfs balance start /mnt/btrfs_disk
```

Важно — При запуске балансировки может возникнуть ошибка (рисунок 31). В таком случае можно использовать флаг `--full-balance`, который отключает предупреждение и сразу запускает балансировку (рисунок 32). Следует учесть, что использование флага `--full-balance` небезопасно, лучше всего использовать фильтры, как указано в пункте **Ошибка! Источник ссылки не найден.** текущей инструкции.

```
live@rosa-ry5ugb ~ $ sudo btrfs balance start /mnt/btrfs_disk
WARNING:

        Full balance without filters requested. This operation is very
        intense and takes potentially very long. It is recommended to
        use the balance filters to narrow down the scope of balance.
        Use 'btrfs balance start --full-balance' option to skip this
        warning. The operation will start in 10 seconds.
        Use Ctrl-C to stop it.
10 9 8 7 6 5 4 3 2 1
Starting balance without any filters.
Done, had to relocate 3 out of 3 chunks
```

Рисунок 31 — Ошибка запуска балансировки

```
live@rosa-ry5ugb ~ $ sudo btrfs balance start --full-balance /mnt/btrfs_disk
Done, had to relocate 3 out of 3 chunks
```

Рисунок 32 — Запуск балансировки с флагом --full-balance

д) запустить балансировку с ограничением по заполнению следующей командой (рисунок 33):

```
sudo btrfs balance start –dusage=50 /mnt/btrfs_disk
```

```
live@rosa-ry5ugb ~ $ sudo btrfs balance start -dusage=50 /mnt/btrfs_disk
Done, had to relocate 1 out of 3 chunks
```

Рисунок 33 — Запуск балансировки с ограничением по заполнению

е) проверить статус балансировки с помощью команды (рисунок 34):

```
sudo btrfs balance status /mnt/btrfs_disk
```

```
live@rosa-ry5ugb ~ $ sudo btrfs balance status /mnt/btrfs_disk
No balance found on '/mnt/btrfs_disk'
```

Рисунок 34 — Проверка статуса балансировки

Примечание — При необходимости можно запустить балансировку только метаданных. Для этого нужно выполнить следующую команду (рисунок 35):

```
sudo btrfs balance start -m /mnt/btrfs_disk
```

```
live@rosa-ry5ugb ~ $ sudo btrfs balance start -m /mnt/btrfs_disk
Done, had to relocate 2 out of 3 chunks
```

Рисунок 35 — Запуск балансировки метаданных

3.5.5.2 Работа с подтомами

Подтома ведут себя как отдельные файловые системы, но используют общее пространство диска.

3.5.5.2.1 Создание подтомов

Для создания корневого подтома с именем "@" нужно выполнить следующую команду (рисунок 36):

```
sudo btrfs subvolume create /mnt/btrfs_disk/@
```

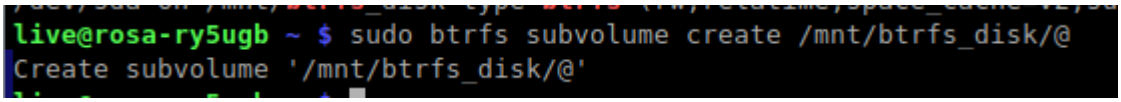


Рисунок 36 — Создание корневого подтома

Чтобы создать подтом @home для хранения пользовательских данных, нужно выполнить команду (рисунок 37):

```
sudo btrfs subvolume create /mnt/btrfs_disk/@home
```

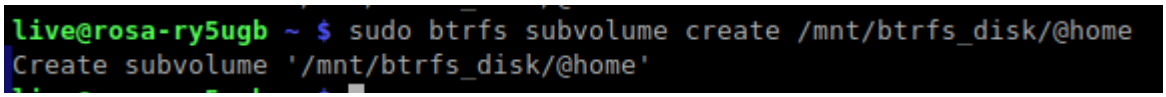


Рисунок 37 — Создание подтома @home

Чтобы создать подтом @data для хранения пользовательских данных отдельно от Системы, нужно выполнить команду (рисунок 38):

```
sudo btrfs subvolume create /mnt/btrfs_disk/@data
```

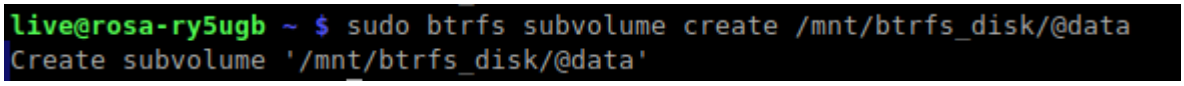


Рисунок 38 — Создание подтома @data

Чтобы просмотреть список подтомов, нужно выполнить следующую команду (рисунок 39):

```
sudo btrfs subvolume list /mnt/btrfs_disk
```

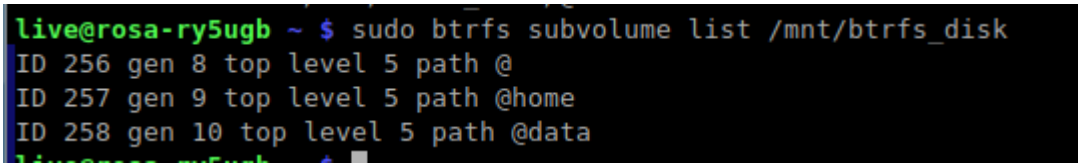


Рисунок 39 — Просмотр списка подтомов

Примечание — Для просмотра детальной информации о подтоме нужно использовать команду (рисунок 40):

```
sudo btrfs subvolume show /mnt/btrfs_disk/@
```



Рисунок 40 — Просмотр информации о подтоме

3.5.5.2.2 Назначение подтома по умолчанию

Подтом по умолчанию (default subvolume) — это механизм BTRFS, который определяет, какой подтом будет монтироваться автоматически, если при монтировании не указан параметр subvol=.

Примечание — Без использования подтома по умолчанию при каждом монтировании нужно указывать подтом. Например, с помощью команды:

```
sudo mount -o subvol=@ /dev/sda /mnt
```

Для назначения подтома "@" подтомом по умолчанию нужно выполнить команду:

```
sudo btrfs subvolume set-default 256 /mnt/btrfs_disk.
```

Для проверки того, что подтом по умолчанию действительно изменён, используется команда (рисунок 41):

```
sudo btrfs subvolume get-default /mnt/btrfs_disk
```

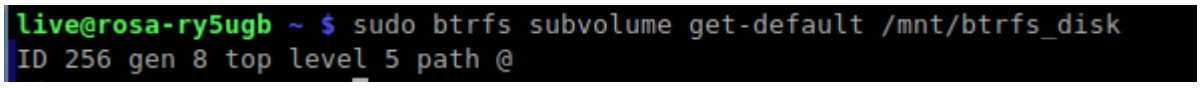


Рисунок 41 — Проверка подтома по умолчанию

Теперь при монтировании без указания подтома с помощью команды `sudo mount /dev/sda /mnt` автоматически будет монтироваться подтом "@".

3.5.5.3 Работа со снапшотами

С помощью снапшота можно сохранить состояние подтома в определенный момент времени и восстановить Систему при необходимости.

3.5.5.3.1 Создание снапшотов

Для создания снапшотов необходимо:

а) создать тестовый файл в подтоме с помощью команды (рисунок 42):

```
echo "Данные для снапшота" | sudo tee /mnt/btrfs_disk/@/snapshot_test.txt
```

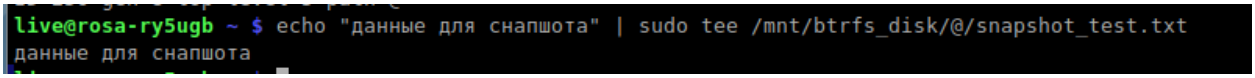


Рисунок 42 — Создание тестового файла в подтоме

б) создать read-write снапшот с помощью команды (рисунок 43):

```
sudo btrfs subvolume snapshot /mnt/btrfs_disk/@ /mnt/btrfs_disk/@snap_rw
```

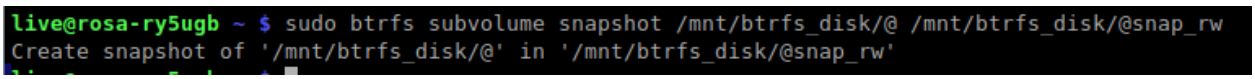


Рисунок 43 — Создание read-write снапшота

в) сделать снапшот доступным только для чтения с помощью команды (рисунок 44):

```
sudo btrfs subvolume snapshot -r /mnt/btrfs_disk/@ /mnt/btrfs_disk/@snap_ro
```

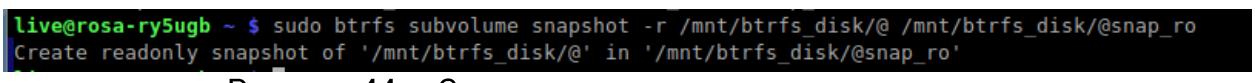


Рисунок 44 — Создание снапшота только для чтения

г) просмотреть список созданных снапшотов с помощью команды (рисунок 45):

```
sudo btrfs subvolume list /mnt/btrfs_disk
```

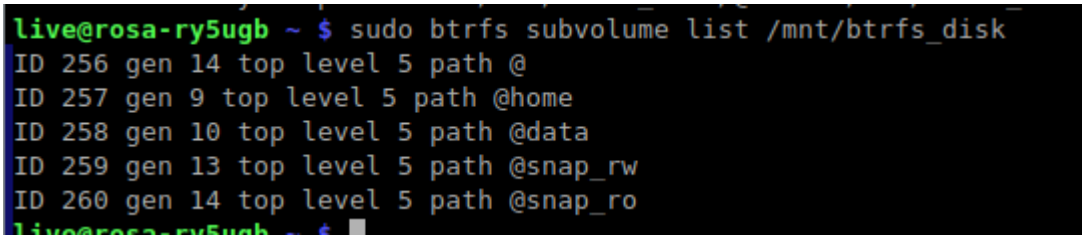


Рисунок 45 — Просмотр списка снапшотов

3.5.5.3.1 Удаление снапшотов

Для удаления снапшота необходимо выполнить следующую команду (рисунок 46):

```
sudo btrfs subvolume delete /mnt/btrfs_disk/@snap_rw
```

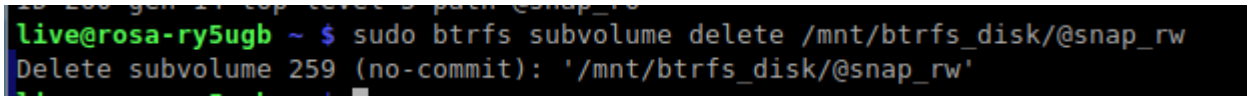


Рисунок 46 — Удаление снапшота

Проверить удаление снапшота можно с помощью команды (рисунок 47):

```
sudo btrfs subvolume list /mnt/btrfs_disk
```

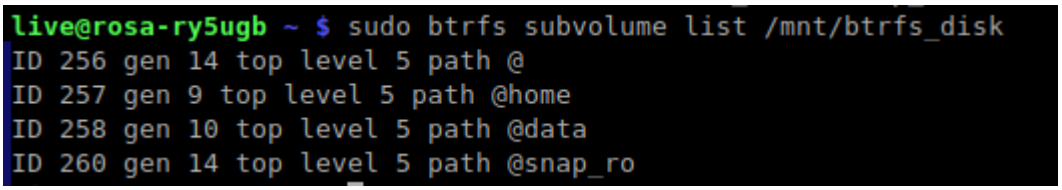


Рисунок 47 — Проверка удаления снапшота

3.5.5.3.2 Восстановление состояния Системы из снапшота

Для проверки восстановления утерянных данных необходимо:

а) создать тестовый файл с помощью команды (рисунок 48):

```
echo "Важные данные" | sudo tee /mnt/btrfs_disk/@/important.txt
```

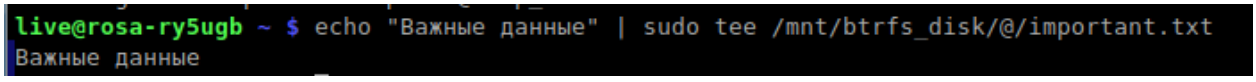


Рисунок 48 — Создание тестового файла для восстановления

б) создать резервный снапшот с помощью команды (рисунок 49):

```
sudo btrfs subvolume snapshot /mnt/btrfs_disk/@ /mnt/btrfs_disk/@backup
```

```
live@rosa-ry5ugb ~ $ sudo btrfs subvolume snapshot /mnt/btrfs_disk/@ /mnt/btrfs_disk/@backup
Create snapshot of '/mnt/btrfs_disk/@' in '/mnt/btrfs_disk/@backup'
```

Рисунок 49 — Создание резервного снапшота

в) удалить созданный файл с помощью команды:

```
sudo rm /mnt/btrfs_disk/@/important.txt
```

г) проверить удаление файла с помощью следующей команды (рисунок 50):

```
ls -la /mnt/btrfs_disk/@/
```

```
live@rosa-ry5ugb ~ $ ls -la /mnt/btrfs_disk/@/
итого 20
drwxr-xr-x 1 root root 34 фев 18 14:12 .
drwxr-xr-x 1 root root 52 фев 18 14:10 ..
-rw-r--r-- 1 root root 37 фев 18 13:52 snapshot_test.txt
```

Рисунок 50 — Проверка удаления файла

д) восстановить файл из снапшота с помощью команды:

```
sudo cp -a /mnt/btrfs_disk/@backup/important.txt /mnt/btrfs_disk/@/
```

е) проверить, что файл действительно восстановлен и содержит исходные данные, с помощью команды (рисунок 51):

```
cat /mnt/btrfs_disk/@/important.txt
```

```
live@rosa-ry5ugb ~ $ cat /mnt/btrfs_disk/@/important.txt
Важные данные
```

Рисунок 51 — Проверка восстановления файла

3.5.5.1 Включение и настройка квот

Квоты в BTRFS позволяют ограничивать использование дискового пространства для подтомов и их групп. В работе с квотами администратор может выполнить следующие действия:

– активировать подсистему поддержки квот на ФС с помощью команды `sudo btrfs quota enable /mnt/btrfs_disk`, иначе квоты не будут работать;

– просмотреть текущие группы квот с помощью следующей команды (рисунок 52):

```
sudo btrfs qgroup show /mnt/btrfs_disk
```

```
live@rosa-ry5ugb ~ $ sudo btrfs qgroup show /mnt/btrfs_disk
Qgroupid    Referenced    Exclusive   Path
--------    ----------    ---------   ----
0/5           16.00KiB     16.00KiB   <toplevel>
0/256         16.00KiB     16.00KiB   @
0/257         16.00KiB     16.00KiB   @home
0/258         16.00KiB     16.00KiB   @data
0/260         16.00KiB     16.00KiB   @snap_ro
0/261         16.00KiB     16.00KiB   @backup
0/262        100.02MiB    100.02MiB   @/@
```

Рисунок 52 — Просмотр текущей группы квот

– создать пользовательскую группу квот с идентификатором 1/100 командой:

```
sudo btrfs qgroup create 1/100 /mnt/btrfs_disk
```

– установить лимит в 500 МБ для группы 1/100 командой:

```
sudo btrfs qgroup limit 500M 1/100 /mnt/btrfs_disk
```

Примечание — Когда суммарный размер данных в группе достигнет лимита, запись будет запрещена и появится ошибка "No space left".

– привязать подтом к группе квот командой:

```
sudo btrfs qgroup assign 0/256 1/100 /mnt/btrfs_disk
```

– просмотреть существующие лимиты и их использование командой (рисунок 53):

```
sudo btrfs qgroup show -re /mnt/btrfs_disk
```

```
live@rosa-ry5ugb ~ $ sudo btrfs qgroup show -re /mnt/btrfs_disk
Qgroupid    Referenced    Exclusive  Max referenced  Max exclusive   Path
--------    ----------    ---------  --------------  -------------   ----
0/5           16.00KiB     16.00KiB            none           none   <toplevel>
0/256         16.00KiB     16.00KiB            none           none   @
0/257         16.00KiB     16.00KiB            none           none   @home
0/258         16.00KiB     16.00KiB            none           none   @data
0/260         16.00KiB     16.00KiB            none           none   @snap_ro
0/261         16.00KiB     16.00KiB            none           none   @backup
0/262        100.02MiB    100.02MiB            none           none   @/@
1/100         16.00KiB     16.00KiB       500.00MiB           none   <0 member qgroups>
```

Рисунок 53 — Просмотр существующих лимитов

– отключить подсистему квот командой:

```
sudo btrfs quota disable /mnt/btrfs_disk
```

Примечание — При отключении подсистемы квот все лимиты перестают действовать, но информация о группах сохраняется. Поэтому при повторном включении настройки восстанавливаются.

3.5.5.2 Графическая программа snapshot-gui

Для более удобного администрирования BTRFS в Системе возможно использование графической утилиты btrfs-snapshot-gui (рисунок 54). С её помощью можно управлять снапшотами без необходимости использования командной строки.

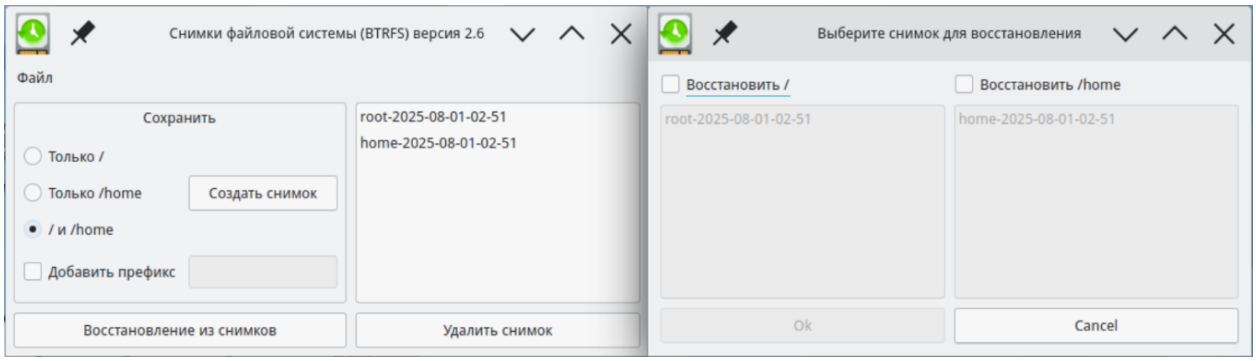


Рисунок 54 — Графическая программа btrfs-snapshot-gui

Для установки пакета btrfs-snapshot-gui необходимо выполнить следующую команду:

```
sudo dnf install btrfs-snapshot-gui
```

Для запуска программы btrfs-snapshot-gui нужно выполнить одно из следующих действия:

– выполнить в консоли команду `btrfs-snapshot-gui`;

– нажать на иконку `Меню запуска приложений` в левом нижнем углу экрана, в строке поиска ввести название программы и нажать клавишу `Enter`.

В приложении доступны:

– создание снимка Системы по кнопке `Создать снимок`;

– удаление снимка Системы по кнопке `Удалить снимок`;

– создание снимка Системы по кнопке `Создать снимок`;

– восстановление Системы из снимка по кнопке `Восстановление из снимков`.

3.5.6 Файловая система ZFS

Файловая система ZFS (Zettabyte File System) — это высокопроизводительная файловая система с поддержкой копирования при записи (COW). ZFS объединяет функции файловой системы и менеджера томов, обеспечивая расширенные возможности по управлению данными, их целостности и оптимизации хранения. Благодаря архитектуре ZFS особенно

эффективна на серверах и в системах хранения данных, где требуются высокая надежность и масштабируемость.

Основные особенности ZFS:

– ZFS использует механизм копирования при записи (COW), что гарантирует целостность данных даже при сбоях питания. Транзакционная модель гарантирует, что файловая система всегда находится в согласованном состоянии;

– Для проверки целостности данных в ZFS используется механизм scrub, который считывает все данные и проверяет их контрольные суммы, автоматически исправляя ошибки при наличии избыточности;

– В репозитории доступны утилиты для автоматизации создания снапшотов и их передачи на удаленные серверы, которые позволяют отправлять только изменения между снапшотами.

3.5.6.1 Начало работы с ZFS

Перед началом работы необходимо убедиться, что в ОС установлена и загружена поддержка файловой системы ZFS. Для этого нужно выполнить команду:

```
lsmod | grep zfs
```

В результате её выполнения будет выведен список загруженных модулей ZFS, если они имеются в Системе (рисунок 55).

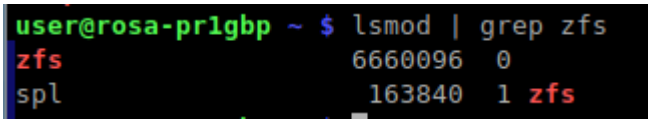


Рисунок 55 — Проверка загруженных модулей ZFS

Примечание — Если модули ФС не загружены, необходимо установить пакеты OpenZFS с помощью команды:

```
sudo dnf install zfs
```

После установки загрузить модули командой (рисунок 56):

```
sudo modprobe zfs
```

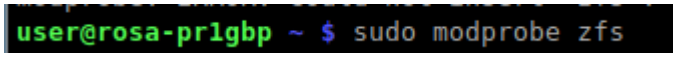


Рисунок 56 — Загрузка модулей ZFS

Полный список всех доступных команд и основных опций для работы с файловой системой ZFS можно посмотреть с помощью команды `zpool help` или `zfs help` (рисунок 57).

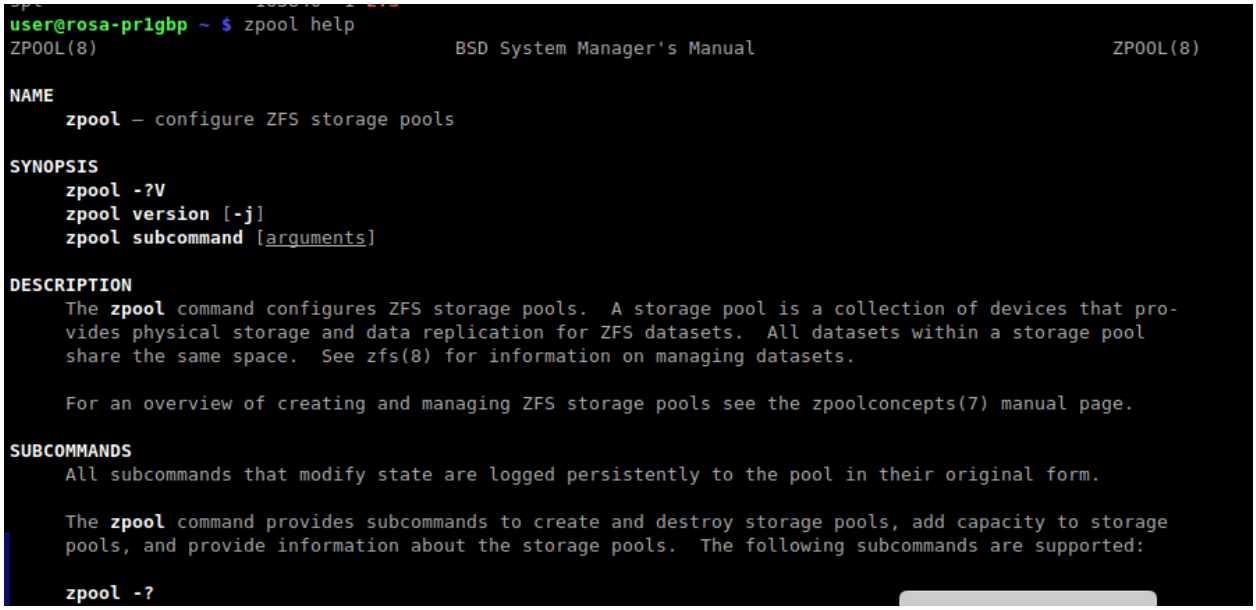


Рисунок 57 — Список всех доступных команд и опций ZFS

3.5.6.2 Создание ZFS-пула и датасетов

ZFS использует двухуровневую архитектуру: пулы хранения (zpools) и датасеты (datasets), которые создаются внутри пулов.

3.5.6.2.1 Создание пула

Пул в ZFS создается командой:

```
sudo zpool create tank /dev/sdb
```

где:

– zpool create — команда для создания пула хранения;
– tank — имя создаваемого пула (может быть любым);
– /dev/sdb — диск, на котором нужно создать пул.

Проверить создание пула можно командой (рисунок 58):

```
sudo zfs list
```

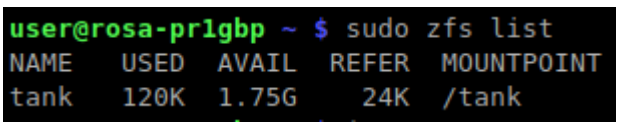


Рисунок 58 — Проверка создания пула

3.5.6.2.2 Создание ФС

Создание файловой системы ZFS выполняется в пуле командой:

```
sudo zfs create tank/data
```

Проверить результат создания ZFS можно командой (рисунок 59):

```
sudo zpool status
```

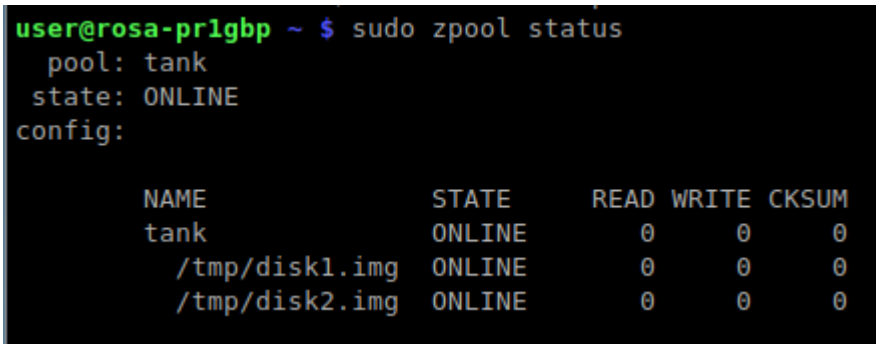


Рисунок 59 — Проверка создания ФС

Проверить версию ZFS можно с помощью команды (рисунок 6019):

```
zfs --version
```

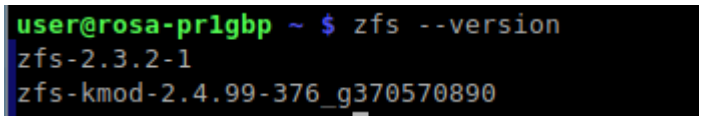


Рисунок 60 — Проверка версии ZFS

Создание ZFS с указанием точки монтирования выполняется командой:

```
sudo zfs create -o mountpoint=/mnt/data tank/data
```

3.5.6.2.3 Создание датасетов

Датасеты в ZFS аналогичны подтомам в BTRFS.

Для создания корневого датасета используется команда:

```
sudo zfs create tank/ROOT
```

Датасет для пользовательских данных создается командой:

```
sudo zfs create tank/HOME
```

Датасет для хранения данных создается командой:

```
sudo zfs create tank/DATA
```

Просмотреть список всех датасетов можно командой (рисунок 61):

```
sudo zfs list
```

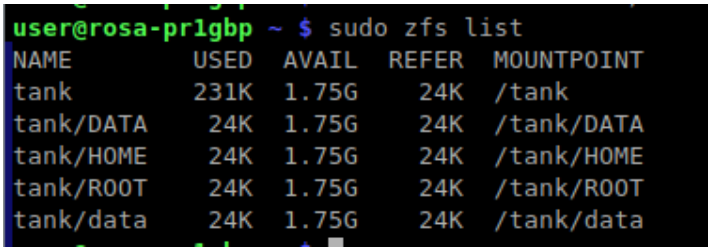


Рисунок 61 — Список всех датасетов

3.5.6.3 Проверка целостности ФС

Для проверки целостности данных в ZFS используется механизм scrub.

Для запуска scrub используется команда:

```
sudo zpool scrub tank
```

Для проверки статуса выполнения механизма scrub выполняется команда (рисунок 62):

```
sudo zpool status
```

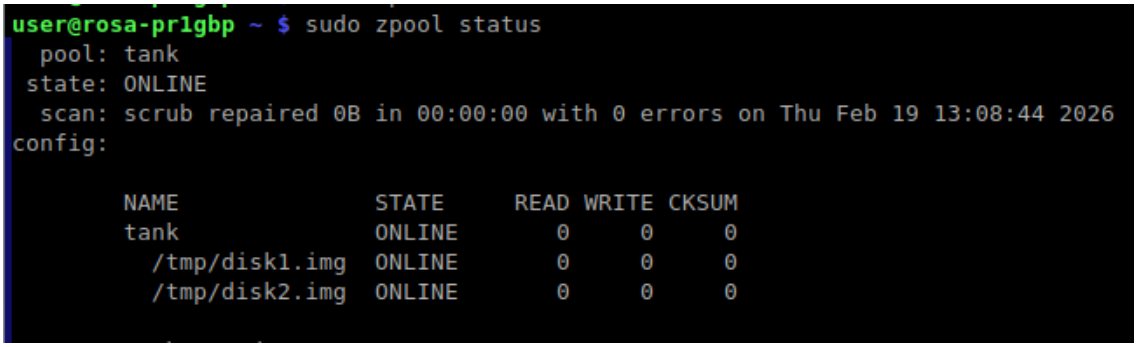


Рисунок 62 — Проверка статуса выполнения механизма скраба

При обнаружении поврежденных блоков ZFS автоматически исправляет их, если пул настроен с избыточностью. В выводе команды `zpool status` строка "scan: scrub repaired 0B" означает, что ошибок не найдено.

3.5.6.4 Ручное монтирование

ZFS-пулы монтируются автоматически при создании и импорте, но при необходимости можно явно указать точку монтирования для датасета командой:

```
sudo zfs set mountpoint=/mnt/data tank/DATA
```

Проверить успешность монтирования можно командой (рисунок 63**Ошибка! Источник ссылки не найден.**):

```
mount | grep tank
```

```
user@rosa-pr1gbp ~ $ mount | grep tank
tank on /tank type zfs (rw,relatime,xattr,noacl,casesensitive)
tank/data on /tank/data type zfs (rw,relatime,xattr,noacl,casesensitive)
tank/ROOT on /tank/ROOT type zfs (rw,relatime,xattr,noacl,casesensitive)
tank/HOME on /tank/HOME type zfs (rw,relatime,xattr,noacl,casesensitive)
tank/DATA on /mnt/data type zfs (rw,relatime,xattr,noacl,casesensitive)
```

Рисунок 63 — Проверка монтирования датасетов ZFS

3.5.6.5 Автоматическое монтирование

ZFS автоматически монтирует все датасеты при импорте пула. Импорт пула можно выполнить по команде:

```
sudo zpool import tank
```

Для автоматического импорта при загрузке Системы нужно выполнить следующие команды:

```
sudo systemctl enable zfs-import-cache.service
sudo systemctl enable zfs-mount.service
```

Просмотреть свойства датасета, включая точку монтирования, можно с помощью команды (рисунок 64):

```
sudo zfs get mountpoint tank/DATA
```

```
user@rosa-pr1gbp ~ $ sudo zfs get mountpoint tank/DATA
NAME        PROPERTY    VALUE       SOURCE
tank/DATA   mountpoint  /mnt/data   local
```

Рисунок 64 — Просмотр точки монтирования датасета

3.5.6.6 Работа со снапшотами

Снапшоты в ZFS позволяют сохранять состояние датасета в определенный момент времени и занимают место только при изменении данных.

3.5.6.6.1 Создание снапшотов

Для создания снапшотов необходимо:

а) создать тестовый файл с помощью команды (рисунок 65):

```
echo "Данные для снапшота" | sudo tee /tank/data/snapshot_test.txt
```

```
user@rosa-pr1gbp ~ $ echo "Данные для снапшота" | sudo tee /tank/data/snapshot_test.txt
Данные для снапшота
```

Рисунок 65 — Создание тестового файла

б) создать снапшот командой (рисунок 66):

```
sudo zfs snapshot tank/DATA@snap1
```



Рисунок 66 — Создание снапшота

в) просмотреть список созданных снапшотов с помощью команды (рисунок 67):

```
sudo sudo zfs list -t snapshot
```

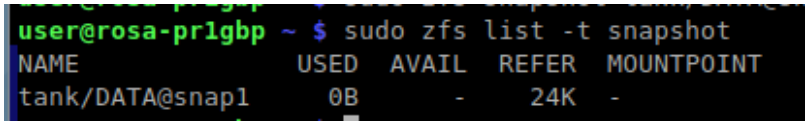


Рисунок 67 — Просмотр списка снапшотов

3.5.6.6.2 Удаление снапшотов

Удаление снапшота выполняется командой (рисунок 68):

```
sudo zfs destroy tank/DATA@snap1
```

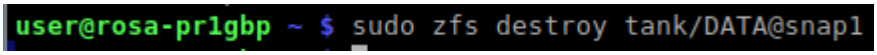


Рисунок 68 — Удаление снапшота

Проверить удаление снапшота можно с помощью следующей команды (рисунок 69):

```
sudo zfs list -t snapshot
```

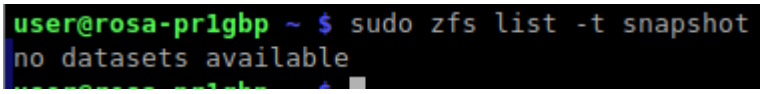


Рисунок 69 — Проверка удаления снапшота

3.5.6.6.3 Восстановление данных из снапшота

Для проверки процедуры восстановления утерянных данных из снапшотов необходимо:

а) создать тестовый файл с помощью команды (рисунок 70):

```
echo "Важные данные" | sudo tee /tank/data/important.txt
```

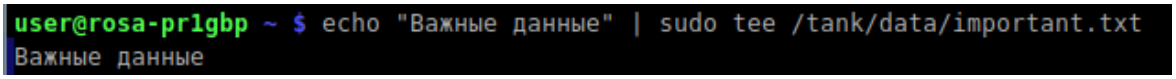


Рисунок 70 — Создание тестового файла для восстановления

б) создать резервный снапшот с помощью команды:

```
sudo zfs snapshot tank/data@backup
```

в) удалить созданный файл с помощью команды:

```
sudo rm /tank/data/important.txt
```

г) проверить удаление файла с помощью команды (рисунок 71):

```
ls -la /tank/data/
```

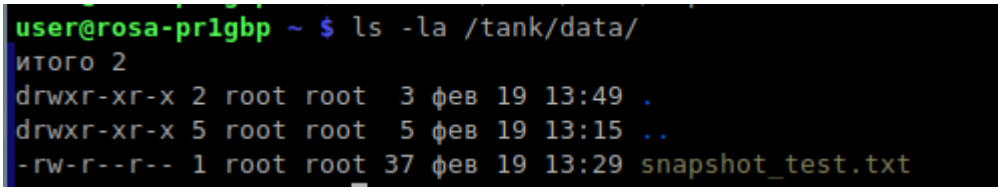


Рисунок 71 — Проверка удаления файла

д) восстановить файл из снапшота с помощью команды:

```
sudo zfs rollback tank/data@backup
```

е) проверить, что файл действительно восстановлен и содержит исходные данные, с помощью команды (рисунок 72):

```
cat /tank/data/important.txt
```

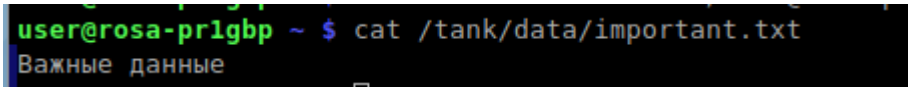


Рисунок 72 — Проверка восстановления файла

3.5.6.7 Включение и настройка квот

ZFS поддерживает квоты на уровне датасетов, позволяя ограничивать использование дискового пространства.

В работе с квотами возможно выполнение следующих действий:

– просмотр текущего использования квот с помощью команды (рисунок 73):

```
sudo zfs list
```

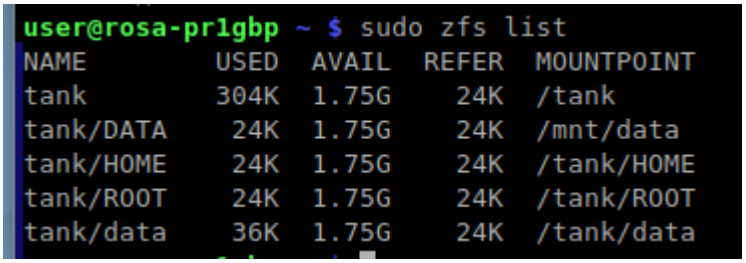


Рисунок 73 — Просмотр текущего использования квот

– установка квоты (например, 500 МБ) для датасета с помощью команды:

```
sudo zfs set quota=500M tank/DATA
```

– просмотр списка установленных квот командой (рисунок 74):

```
sudo zfs get quota tank/DATA
```

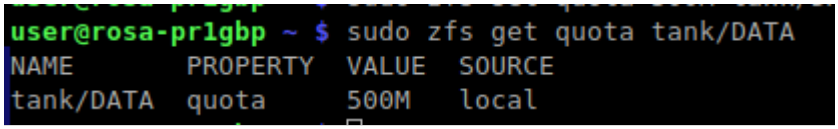


Рисунок 74 — Список установленных квот

– установка ограничения максимального объёма данных, например, в 500 МБ для tank/DATA), с помощью параметра refquota:

```
sudo zfs set refquota=500M tank/DATA
```

– просмотр использования дискового пространства с учетом квот выполняется с помощью команды (рисунок 75):

```
sudo zfs get all tank/DATA | grep -E "quota|used"
```

где:

– tank/DATA – раздел для которого выполняется проверка.

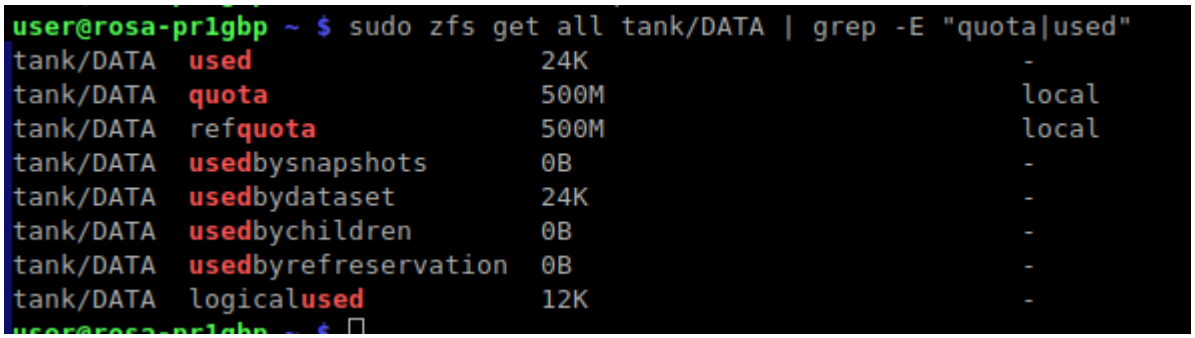


Рисунок 75 — Просмотр существующих лимитов

– снятие установленных квот выполняется командой:

```
sudo zfs set quota=none tank/DATA
```

3.5.7 Файловая система GFS2

Файловая система GFS2 (Global File System 2) является кластерной распределённой файловой системой, которая обеспечивает одновременный доступ к одному и тому же хранилищу с нескольких узлов под управлением ОС на базе Linux. Это решение позволяет организовать совместную работу серверов в кластере с общей файловой структурой при сохранении целостности данных и согласованности доступа.

ОС включают поддержку GFS2 в составе ядра и репозитория пакетов, что позволяет использовать данную файловую систему без необходимости сборки модулей вручную.

GFS2 использует модель общего доступа к блочному устройству (Shared Block Storage) с распределённым управлением блокировками через DLM (Distributed Lock Manager). Все узлы в кластере обнаруживают один и тот же том, а механизм DLM обеспечивает согласование одновременного чтения и записи.

Основные особенности GFS2:

– согласованное использование метаданных между узлами;
– управление журналами транзакций;
– возможность расширения размера файловой системы;
– работа поверх общих хранилищ, подключаемых через SAN, iSCSI или Fibre Channel.

Для координации работы нескольких серверов используется кластерный менеджер (Pacemaker/Corosync), который отвечает за согласование конфигурации кластера и мониторинг доступности узлов.

Для работы с GFS2 требуется установить необходимые утилиты пользовательского пространства с помощью команды:

```
sudo dnf install gfs2-utils
```

Создание файловой системы GFS2 на общем устройстве выполняется командой:

```
sudo mkfs.gfs2 -p lock_dlm -t cluster_fs /dev/sdb
```

где:

– lock_dlm — механизм блокировок DLM;
– cluster_fs — имя кластера, которое должно соответствовать конфигурации Pacemaker/Corosync;
– /dev/sdb — общий раздел или блочное устройство.

Для монтирования файловой системы GFS2 на сервере выполняются следующие действия:

```
sudo mkdir -p /mnt/gfs2
sudo mount -t gfs2 /dev/sdb /mnt/gfs2
```

На клиентских узлах монтирование выполняется аналогично, с указанием сетевого имени сервера:

```
sudo mount -t gfs2 server1:/mnt/gfs2 /mnt/gfs2
```

где server1 — DNS-имя или IP-адрес сервера.

Для обеспечения корректной синхронизации узлов в кластере используется связка Pacemaker и Corosync. Необходимо:

– настроить Corosync для обмена сообщениями между узлами;
– сконфигурировать Pacemaker для управления ресурсом файловой системы (в том числе fencing, если требуется);
– зарегистрировать ресурс в Pacemaker, указав имя кластера, совпадающее с параметром -t при создании ФС.

Для мониторинга состояния GFS2 и диагностики возможных ошибок следует использовать следующие инструменты:

– journalctl — просмотр системных сообщений и логов ядра;
– gfs2_tool — проверка и получение состояния ФС;
– gfs2_quota — управление квотами в GFS2;
– fsck.gfs2 — проверка целостности файловой системы (применяется только при отключённом кластере).

Также рекомендуется контролировать загрузку сети и задержки, так как GFS2 критично зависит от стабильности межузловых соединений.

3.5.8 Указатели

Ссылка на файл в ОС – это указатель на файл. Если проводить аналогию с ОС Windows, то ссылки схожи с ярлыками. То есть при создании ссылки, которая указывает на какой-либо файл или директорию, можно разместить эту ссылку в другом каталоге. Используя такую ссылку, обращение будет направляться к настоящему файлу или каталогу.

Ссылки в ОС бывают двух типов: **символические** и **жесткие**, которые 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 копию самого файла, на которую она указывает. Она является лишь указателем на файл, однако так же,

как и файл обладает собственными правами доступа (которые не распространяются на целевой файл) и содержит путь до целевого файла. Можно создавать несколько символических ссылок на один файл и эти ссылки могут иметь разные имена.

Связь между символической ссылкой и файлом, на который она указывает, является "мягкой". Если удалить символическую ссылку, то файл, на который она указывает, не удаляется.

Жесткая ссылка (hard link) является своего рода синонимом для существующего файла. При создании жесткой ссылки создается дополнительный указатель на существующий файл, но не копия файла.

Жесткие ссылки выглядят в файловой структуре как еще один файл. Если жесткая ссылка создается в том же каталоге, где находится целевой файл, то они должны иметь разные имена. Жесткая ссылка на файл должна находится в той же ФС, что и другие жесткие ссылки на этот файл.

В ОС каждый файл имеет уникальный идентификатор индексный дескриптор (inode) – это число, которое однозначно идентифицирует файл в ФС. 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.

Особенности жестких ссылок:

– нельзя создавать для директорий;
– не может указывать на несуществующий файл;
– редко используются в повседневной работе.

3.5.8.1 Создание и удаление ссылок

Для создания ссылок в ОС используется команда `ln`.

Синтаксис команды `ln`:

```
ln опции целевой_файл имя_ссылки
```

Обычно используется только одна опция `-s`. Полный список опций можно получить, выполнив следующую команду:

```
man ln
```

Чтобы создать символическую ссылку, нужно выполнить команду `ln` с опцией `-s`:

```
ln -s целевой_файл имя_символьной_ссылки
```

Чтобы создать жесткую ссылку нужно использовать команду `ln` без опции `-s`.

```
ln целевой_файл имя_жесткой_ссылки
```

Ссылки, как и обычные файлы, можно удалять, используя команду `rm`:

```
rm имя_ссылки
```

3.5.9 Безопасное удаление файлов

Для предотвращения восстановления удалённой информации операционная система предоставляет утилиту Rosa Shred, выполняющую многократную перезапись содержимого файла перед его удалением. Такая процедура снижает вероятность восстановления данных с носителей, особенно на классических HDD.

Следует учитывать, что полное гарантированное стирание данных на SSD-носителях не всегда достижимо из-за особенностей внутреннего управления памятью. В таких случаях рекомендуется использовать методы аппаратного сброса накопителей или шифрование данных перед использованием.

Установка утилиты Rosa Shred производится из встроенного репозитория командой:

```
sudo dnf install rosa-shred
```

3.5.9.1 Графическая оболочка Rosa Shred

Утилита Rosa Shred работает от имени пользователя, под учетной записью которого она была запущена. Таким образом, утилита может удалить лишь те файлы или каталоги, на которые у данного пользователя есть права, достаточные для удаления.

Утилита ROSA Shred несколько раз перезаписывает указанные файлы специальными битовыми последовательностями, чтобы сделать более трудоемким процесс восстановления данных даже в случае использования специального оборудования.

Для запуска утилиты необходимо перейти в главное меню ОС "Приложения → ROSA Shred" или выполнить команду `rosa-shred`.

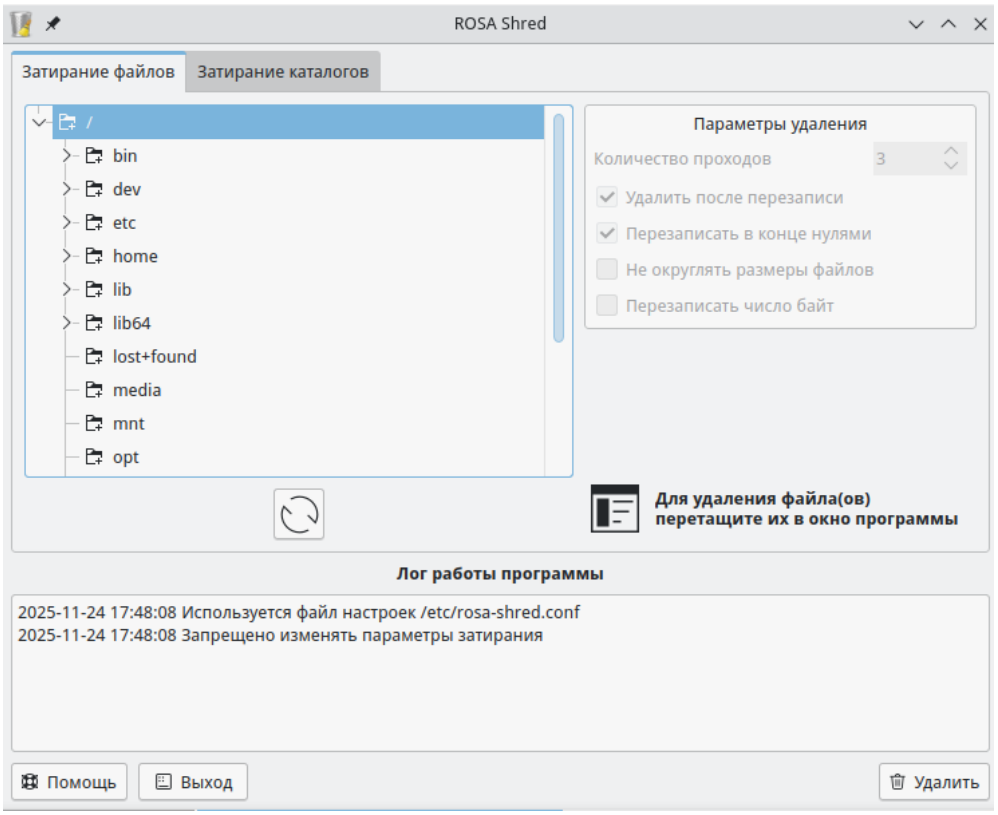


Рисунок 76 – Интерфейс Rosa Shred

В окне слева (рисунок 76) представлена корневая файловая система ОС. Справа представлены опции удаления, которые можно установить для файлов или каталогов.

Опции перезаписи:

– Количество проходов=N – переписать N раз вместо указанных (3) по умолчанию;

– Перезаписать в конце с нулями – перезаписать в конце с нулями, чтобы скрыть перемешивание;

– Удалить после перезаписи – перемешать и удалить файл после перезаписи;

– Не округлять размеры файлов – не округлять размеры файлов до следующего целого блока; по умолчанию – для нерегулярных файлов;

– Перезаписать число байт=N – jчистить N байт, а не файл целиком.

3.5.9.2 Консольная утилита shred

Безопасное удаление файлов в консольном режиме осуществляется с помощью утилиты shred, которая позволяет перезаписать файл случайными данными несколько раз и затем удалить его. Основная форма команды:

```
shred -u имя_файла
```

Основные ключи утилиты:

– -u — перезаписать файл и удалить его после завершения операции;

– -n <число> — задать количество дополнительных циклов перезаписи (по умолчанию выполняется 3 цикла);

– -v — отображать ход выполнения операции;

– -z — выполнить финальную перезапись нулями, чтобы скрыть факт использования механизма безопасного удаления;

– -f — принудительное изменение прав доступа к файлу, если это необходимо для выполнения операции;

– -s <размер> — ограничить объём данных для перезаписи указанным размером (в байтах);

– --random-source=<файл> — использовать указанный источник случайных данных вместо системного генератора;

– --remove=<метод> — задать способ удаления файла после перезаписи (например, unlink или wipe).

Например:

```
shred -u документ.txt
shred -u -n 5 имя_файла.dat
```

3.6 Установка и удаление программ (репозитории)

Для установки, удаления и обновления программ в ОС используется пакетная модель. Программное обеспечение поставляется в виде пакетов стандартного формата (для данной ОС – это rpm), содержащих исполняемые файлы и вспомогательные данные. Менеджер пакетов проверяет наличие необходимых зависимостей и регистрирует приложение в среде (доступ из командной строки и в меню графической оболочки).

Так как ОС построена по модели совместного использования кода, программы в ней гораздо меньше дублируются и более взаимозависимы. Это с одной стороны приводит к уменьшению занимаемого ОС пространства на диске, а с другой это может вызвать проблемы совместимости при установке программ, собранных для других дистрибутивов Linux.

В качестве основной системы управления пакетами используется dnf. Dnf работает с репозиториями, автоматически разрешает зависимости, проверяет подписи и целостность пакетов и тем самым снижает риск нарушений согласованности. Низкоуровневые операции выполняются через rpm и отражаются в базе установленных пакетов.

В ОС программы устанавливаются из собственных централизованных сетевых хранилищ, называемых репозиториями. Каждый репозиторий содержит инсталляционные пакеты, связанные друг с другом зависимостями.

Для обеспечения безошибочной и безопасной работы Системы рекомендуется ставить пакеты только из официальных репозиториев РОСА, предназначенных именно для этой версии.

Репозитории в ОС требуют ключа для их использования, подробнее о размещении ключа в Системе см. п. 3.10.1.

Также для управления подключением онлайн-репозиториев служит файл /etc/dnf/vars/enable_online_repos. Установленное значение "1" в этом файле включает сетевые репозитории, "0" — выключает.

Сетевые репозитории для данной версии ОС по умолчанию подключены.

Для этого нужно выполнить команды:

```
sudo mkdir /mnt/disk1
sudo mount /dev/cdrom /mnt/disk1
```

После этого работа с дисковым репозиторием будет аналогична работе с сетевым. Файл его подключения можно найти по адресу /etc/yum.repos.d/disk1.repo, для его включения нужно установить в этом файле enabled=1.

3.6.1 Использование rpm

Rpm — это формат пакетов программного обеспечения и утилита низкого уровня для управления установленными пакетами в ОС. Инструмент обеспечивает установку и удаление отдельных пакетов, ведение базы установленных компонентов, запрос сведений о пакетах и проверку целостности файлов.

Rpm не управляет репозиториями и не подбирает зависимости пакетов автоматически, она используется как вспомогательный инструмент для диагностики и контроля установленного ПО.

Штатная установка, удаление и обновление программ выполняется высокоуровневым менеджером пакетов dnf. Все операции dnf (установка, обновление, удаление) выполняются через библиотеки rpm и фиксируются в базе rpm.

Типовые запросы rpm:

– Определение, каким пакетом установлен файл:

```
rpm -qf /путь/к/файлу
```

– Перечень файлов, входящих в пакет:

```
rpm -ql имя_пакета
```

– Проверка целостности файлов пакета:

```
rpm -V имя_пакета
```

3.6.2 Файловый менеджер dnf

Dnf — менеджер пакетов высокого уровня, который используется в ОС. Dnf обеспечивает установку, обновление и удаление программ из репозиториев. При работе автоматически подбираются и устанавливаются необходимые зависимости, что упрощает администрирование и снижает риск ошибок. Инструмент рекомендуется как основной для повседневного управления программным обеспечением и имеет графическую оболочку в виде программы "Пакеты (dnfdragora)", доступ к которой осуществляется через главное меню приложений Системы.

Перед установкой dnf выполняет проверку подлинности пакетов (криптографические подписи), что помогает защититься от подмены. Изменения применяются по принципу "целиком или никак": если на любом этапе возникает ошибка (например, не найдена зависимость), установка не выполняется, и Система остаётся в прежнем рабочем состоянии. Выполненные операции можно просматривать в истории, а ход работы фиксируется в журналах менеджера (например, /var/log/dnf.log).

Синтаксис команд dnf:

```
dnf <действие> <пакет> <ключ>
```

Основные команды dnf:

– Установка пакетов:

```
sudo dnf install <имяпакета>
```

После выполнения команды Система автоматически загрузит и установит выбранный пакет, а также все его зависимости.

– Удаление пакетов:

```
sudo dnf erase <имяпакета>
```

Эта команда удаляет не только выбранный пакет, но и все его зависимости.

– Поиск пакетов в Системе:

```
dnf search <ключевое_слово>
```

Эта команда позволяет найти программу или узнать, какие пакеты доступны в репозиториях по ключевому слову, например, browser.

– Просмотр информации о пакете:

```
dnf info <название_пакета>
```

– Очистка кеша и ненужных пакетов:

```
sudo dnf autoremove
sudo dnf clean all
```

Полный список команд dnf доступен в консоли по команде:

```
man dnf
```

3.6.3 Установка программ из сторонних источников

Следует обратить внимание, что установка программ из сторонних репозиториев **не рекомендуется и может нарушить целостность и безопасность Системы**, однако при необходимости она возможна.

Лучшим способом установки программ из сторонних репозиториев будет самостоятельная сборка пакетов из исходных текстов, пользуясь онлайн-средой сборки abf.io (приглашение для регистрации в среде можно получить по запросу к сотрудникам компании-разработчика ОС РОСА). При этом администратор становится владельцем своего отдельного репозитория и сможет с помощью инфраструктуры РОСА собирать требуемые пакеты. Данный способ является самым желательным и распространенным при необходимости установки программ, отсутствующих в репозиториях РОСА.

Установка сторонних программ также возможна и с помощью использования пакетов, созданных для схожих с ОС РОСА "ХРОМ" дистрибутивов. Наиболее совместимыми с пакетами РОСА являются пакеты для OpenMandriva, Mageia и Fedora: их можно попробовать установить командой:

```
sudo dnf install <ФайлПакета>
```

Возможно, зависимости таких пакетов не будут совпадать с зависимостями РОСА и блокировать установку – в таком случае можно попробовать установить скачанный пакет командой:

```
sudo rpm -i --nodeps <ФайлПакета>
```

Такой способ часто используется для установки сторонних драйверов.

Также возможно установить автономные пакеты flatpack или appimage, которые полностью поддерживаются в ОС в качестве систем установки.

3.6.4 Использование Notamock для сборки пакетов

Notamock представляет собой набор утилит для создания изолированных окружений при сборке RPM-пакетов в дистрибутивах ОС из семейства РОСА. Название отражает его сходство с утилитой mock, но при этом реализованы принципиальные отличия — использование многослойных контейнеров с общей базовой корневой файловой системой (rootfs) аналогично технологии Docker. Это позволяет ускорить сборку за счет того, что минимальное окружение всегда готово, а устанавливаются только необходимые зависимости.

Основные компоненты Notamock:

– notamock — утилита для запуска локальной сборки одного проекта;
– notamocks — утилита для массовой (параллельной) локальной сборки нескольких проектов;
– notamockc — утилита для запуска контейнера с любой командой (по умолчанию – bash);
– notamockd — утилита для создания сборочных контейнеров для системы ABF;
– notamocka — скрипт запуска ABF Builder внутри контейнера (устанавливается из пакета notamock-abf).

Все утилиты используют общий конфигурационный файл /etc/notamock.cfg, а также дополнительные настройки из каталога /etc/notamock.d/. При необходимости может быть указан иной конфигурационный файл с параметром --config. Все утилиты требуют прав суперпользователя (root) из-за операций с монтированием.

Базовая корневая файловая система (rootfs) создается для каждой поддерживаемой платформы только один раз и затем используется многократно. Количество одновременно работающих контейнеров, процессов notamock и builders не влияет на количество rootfs. Для пересоздания базовой rootfs необходимо запустить любую утилиту с параметром -u в момент, когда rootfs не используется другими процессами.

При этом для актуализации rootfs достаточно обновления самой Системы, дополнительных действий не требуется.

Использование утилит Notamock:

– **notamock**

Утилита notamock применяется для локальной сборки одного проекта. В типичном сценарии взаимодействует с ABF (Automated Build Farm), но также может пересобрать пакет по src.rpm. Пример пересборки проекта lsof:

```
sudo dnf install abb notamock
abb clone lsof
```

```
cd lsof
notamock
```

При первом запуске будет создана базовая rootfs, что занимает больше времени. Последующие сборки выполняются значительно быстрее.

Полезные параметры notamock:

– -v – подробный вывод сообщений;
– -r – указание целевой платформы (например, -r 13);
– -e – остановка в контейнере перед запуском сборки (удобно для отладки);
– RPMSAVE=yes – сохранение всех скачанных сборочных зависимостей во внутреннем репозитории notamock;
– REUSABLE=yes – повторное использование контейнера для ускорения последовательных сборок одного проекта;
– PM_config – возможность подмены конфигурации пакетного менеджера, например, PM_config=/home/user/dnf.conf;
– TMPD – указание собственного каталога вместо /tmp/notamock.

Все параметры могут быть переопределены напрямую через командную строку или настроены в /etc/notamock.cfg. Более подробную справку можно получить по команде `notamock --help`.

– **notamocks**

Утилита notamocks предназначена для массовых сборок сразу нескольких проектов в параллельном режиме. Предполагается, что проекты располагаются в подкаталогах внутри одного каталога.

Пример запуска утилиты по команде:

```
cd /путь/к/проектам
notamocks
```

Полезные параметры notamocks:

– -p – количество параллельных сборок;
– -s – пропускать уже собранные проекты;
– -n (или --chain) – цепочная сборка с добавлением результатов предыдущих итераций в репозиторий;
– -l – каталог для хранения логов.

– **notamockd**

Утилита notamockd предназначена для интеграции со сборочной системой ABF. Может запускаться в двух режимах:

– создание и запуск контейнера с builders напрямую;

– создание Docker-образа и запуск контейнера через Docker.

Для запуска в основном режиме необходимо указать токен сборщика:

```
notamockd -t <TOKEN>
```

Полезные опции утилиты notamockd:

– -p – количество builders;
– -b – имя builder;
– -B – внешний каталог для сборки;
– --watch – мониторинг логов;
– --tmux – запуск в tmux внутри контейнера для отладки.

Если не указан внешний каталог через опцию -B, сборка будет происходить в /tmp контейнера. В случае необходимости можно изменить поведение на использование chroot опцией CHROOT=INTERNAL в конфигурации или указать TMPD с собственным путем.

Если требуется создание Docker-образа с builder используются следующие команды:

```
sudo dnf install docker
sudo systemctl start docker
notamockd --docker -b builder_name
docker image ls
docker run -it --rm --privileged -e BUILD_TOKEN="ABF_TOKEN"\
-e PARALLEL=3 -v /tmp:/tmp <docker image ID>
```

Примечание – Для корректной работы требуется смонтировать том через -v, иначе overlayfs внутри контейнера не будет работать.

– **notamockc**

Утилита notamockc используется для запуска контейнера с любыми командами в среде chroot, основанной на слоеных rootfs. Это позволяет получить доступ к окружению сборки без необходимости отдельного скачивания образа с ABF или создания rootfs вручную.

Для запуска утилиты достаточно выполнить команду:

```
notamockc
```

При необходимости можно указать целевую платформу с помощью параметра -r, например:

```
notamockc -r 13
```

Также возможно сразу передать команду, которая будет выполнена в chroot:

```
notamockc -r 13 -- dnf provides thunderbird
```

Если требуется сохранить изменения между запусками, следует создать именованный контейнер:

```
notamockc -n my_container
```

Чтобы изменения контейнера сохранялись и между перезагрузками хоста, необходимо указать путь для хранения временных данных через TMPD как в командной строке, так и в конфигурационном файле /etc/notamock.cfg.

Для проброса необходимых файлов в контейнер удобно использовать параметр BIND. Например:

```
notamockc BIND='/home/user/project::/root/project'
```

Также можно указать дополнительные опции монтирования, например, только для чтения и без возможности исполнения:

```
notamockc
BIND='/home/user/project::/root/project::ro::noexec'
```

Помимо каталогов, возможно монтирование squashfs-архивов, образов или разделов диска. В данном случае происходит полноценное монтирование, а не bind. Опция bind доступна как в командной строке, так и через конфигурационные файлы, и может применяться во всех утилитах Notamock.

Notamock может применяться для сборки пакетов с последующим статическим анализом кода с помощью комплекса SVACE. Для этого нужно выполнить следующие шаги:

а) На хостовой Системе распаковать архив SVACE по команде, например:

```
/home/user/svace/svace-3.4.240312-x64-linux
```

Создать файл конфигурации пакетного менеджера, например, /etc/notamock-dnf.conf:

```
[main]
gpgcheck=0
best=True
skip_if_unavailable=False
keepcache=True
reposdir=''

[rosa-contrib]
name=rosa-contrib
```

```
    baseurl=http://abf-
downloads.rosalinux.ru/rosa$releasever/repository/$arch/contrib/
release/
    gpgcheck=0

    [rosa-main]
    name=rosa-main
    baseurl=http://abf-
downloads.rosalinux.ru/rosa$releasever/repository/$arch/main/re\
lease/
    gpgcheck=0
```

Создать отдельный профиль Notamock (например, /etc/notamock.d/svace) командой:

```
    PM_config=/etc/notamock-dnf.conf
    BUILD_cmd='/opt/svace/bin/svace init && /opt/svace/bin/svace
build rpmbuild -ba *.spec --define="_sourcedir $INTERNAL_DIR" --
define="_topdir $INTERNAL_DIR"'
    BIND_SOURCE=yes
    BIND='/home/user/svace/svace-3.4.240312-x64-
linux:/opt/svace:ro'
```

Для запуска сборки выполнить команду:

```
    sudo notamock -r 2021.15 name-X-Y.src.rpm
```

Результаты анализа будут сохранены в каталоге name/.svace-dir относительно текущей директории запуска. При необходимости последующего анализа эти данные можно перенести на другую машину и использовать утилиту svace analyze.

3.6.5 Архивный репозиторий пакетов repo-archive

В ОС наряду с основными репозиториями доступен архивный репозиторий пакетов repo-archive, который предназначен для хранения всех ранее существовавших версий пакетов, которые когда-либо входили в официальные репозитории ОС , но были исключёны из основного репозитория в процессе обновлений.

Использование архивных репозиториев является типичной практикой для Linux-дистрибутивов и применяется для обеспечения воспроизводимости, поддержки устаревших конфигураций и восстановления проверенных окружений.

Архивный репозиторий может использоваться в следующих типовых сценариях:

– развёртывание новых серверов или рабочих станций с проверенной и ранее используемой версией ПО;

– поддержка инфраструктурных компонентов, требующих фиксированной версии;

– восстановление Системы в конфигурации, идентичной ранее эксплуатируемой;

– анализ и отладка проблем, возникших после обновления пакетов.

Пример сценария применения repo-archive: на предприятии используется платформа контейнеризации Kubernetes версии 1.20. В официальном репозитории ОС пакет обновляется до версии 1.30, однако переход на новую версию ещё не выполнен. Для развёртывания дополнительного сервера в существующей инфраструктуре требуется установка Kubernetes версии 1.20. В этом случае необходимая версия пакета может быть установлена именно из репозитория repo-archive.

При работе с архивным репозиторием необходимо учитывать следующие ограничения:

– совместимость старых версий пакетов с актуальными версиями других пакетов не гарантируется;

– использование архивных версий может потребовать фиксации зависимостей и дополнительных проверок;

– избегать смешивания архивных и актуальных пакетов без предварительного тестирования.

Для подключения архивного репозитория в Системе необходимо установить пакет командой:

```
sudo dnf install rosa-repos-archive
```

После установки пакета репозиторий repo-archive автоматически добавляется в конфигурацию пакетного менеджера dnf и может использоваться для установки пакетов и разрешения зависимостей.

В случае возникновения проблем, связанных с доступом к архивному репозиторию или его содержимым, следует обращаться в службу поддержки по адресу электронной почты: mirror-support@rosa.ru.

3.7 Управление жесткими дисками

Правильное управление и обслуживание жестких дисков играют ключевую роль в обеспечении надежности и производительности Системы, а также в защите данных от потерь. Эффективное управление дисковыми устройствами

напрямую влияет на производительность Системы. Помимо физических дисков, Система может работать с виртуальными дисковыми устройствами, которые используются в контейнеризации, виртуализации и других технологиях.

Раздел диска – это часть долговременной памяти накопителя данных, логически выделенная для удобства работы и состоящая из смежных блоков. Информация о разделах хранится в таблице разделов.

Все физические устройства памяти в Системе располагаются в каталоге /dev. Каждому из дисков по умолчанию присваивается последовательное название: /dev/sdX, где X — последовательные буквы a,b,c,d,e,... в зависимости от порядкового номера диска.

Обычно каждый раздел диска рассматривается ОС как отдельный логический диск, даже если все они находятся на одном физическом носителе. В ОС каждый раздел назначается каталогу /dev, например, /dev/sda1 или /dev/sda2 и так далее.

Существует два основных способа хранения информации о разделах на жестких дисках. Первый MBR (основная загрузочная запись), а второй GPT (таблица разделов GUID).

Разбиение диска на разделы также происходит и в ходе инсталляции Системы на шаге выбора места установки ОС.

3.7.1 Fdisk

Стандартной утилитой для управления разделами MBR в ОС является fdisk – это интерактивная утилита с меню. С ее помощью можно создавать, изменять, удалять и просматривать разделы дисков.

Все операции с дисками необходимо выполнять от имени пользователя root или с привилегиями root, используя команду sudo.

Для просмотра списка доступных дисков и разделов используются команды:

```
lsblk
```

или

```
disk -l
```

Важно – Для начала работы с fdisk необходимо открыть терминал и выполнить команду, указав путь к устройству, с которым требуется работать:

Основные команды для работы с fdisk:

```
sudo fdisk /dev/sda
```

где /dev/sda — это имя устройства, соответствующего вашему жесткому диску. После перехода к жесткому диску вводят следующие параметры команды:

– m – выводит список всех доступных команд в fdisk и используется для полного ознакомления с утилитой;

– p – выводит на экран текущую таблицу разделов, чтобы увидеть, какие разделы уже созданы на диске;

– n – создать новый раздел: система предложит выбрать тип раздела (основной или расширенный) и указать начальный и конечный сектора;

– d – удаляет выбранный раздел: рекомендуется использовать с осторожностью, так как удаление раздела приведет к потере всех данных на нем;

– w – сохраняет все изменения, сделанные в таблице разделов, и завершает работу fdisk. После записи изменений они вступают в силу и их нельзя отменить;

– q – выйти без сохранения изменений, сделанных в текущем сеансе;

– t – изменение типа раздела, за которым следует номер раздела, подлежащий изменению.

3.7.1.1 Первичные и расширенные разделы

На MBR-диске может быть 2 основных типа разделов: основной первичный (primary) и расширенный (extended). Если требуется сделать диск "загрузочным", первый раздел должен быть основным. Следует учесть, что в таблице MBR может быть только 4 основных раздела на диске.

Один из способов обойти это ограничение – создать расширенный раздел, который действует как контейнер для логических разделов. Например, могут быть следующие разделы: основной, дополнительный, занимающий оставшуюся часть дискового пространства и пять логических разделов внутри него.

В ОС первичный и расширенный разделы обрабатываются одинаково, поэтому нет никаких преимуществ использования одного над другим.

3.7.1.2 Управление разделами

Создание нового раздела:

а) запуск fdisk для нужного диска:

```
sudo fdisk /dev/sda
```

б) чтобы создать новый раздел, необходимо ввести команду n (new);

в) выполнить инструкции утилиты, указав:

– какой тип раздела необходимо создать: основной (primary) или расширенный (extended).

Примечание – Обычно используется основной.

– номер раздела (например, номер "1" для первого раздела);

– начальный сектор. Если оставить поле пустым, будет использован первый доступный сектор;

– конечный сектор или размер раздела (например, +10Гб для 10 Гб);

г) если всё прошло успешно, ввести параметр p, чтобы просмотреть таблицу разделов и убедиться, что новый раздел создан;

д) для того чтобы записать изменения и выйти необходимо ввести команду `w`.

Удаление раздела:

а) запустить fdisk для нужного диска:

```
sudo fdisk /dev/sda
```

б) для того чтобы удалить раздел необходимо ввести команду `d`;

в) утилита предложит выбрать номер раздела, который необходимо удалить;

г) ввести команду `p`, чтобы убедиться, что раздел удален;

д) ввести команду `w` для того, чтобы записать изменения и выйти.

Примечание – Перед внесением любых изменений в разделы диска рекомендуется сделать резервную копию всех важных данных, так как изменения в таблице разделов могут привести к потере данных.

3.7.1.3 Проверка нераспределенного пространства диска

Если неизвестно сколько свободного места осталось на диске, то можно использовать команду `F(free)`, чтобы показать нераспределенное пространство, например:

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

 start End Sectors Size
2099200 3903577 1804378 881M
```

3.7.2 Gdisk

Для работы с GPT-дисками применяется утилита gdisk, являющаяся функциональным аналогом fdisk, но адаптированная для GUID-таблиц разделов. Её интерфейс интерактивный и использует похожие команды.

3.7.2.1 Просмотр таблицы разделов

Для отображения текущей таблицы разделов используется команда `p`. Пример вывода:

```
Command (? for help): p

Disk /dev/sdb: 3903578 sectors, 1.9 GiB
Model: DataTraveler 2.0
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): AB41B5AA-A217-4D1E-8200-E062C54285BE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 3903544
Partitions will be aligned on 2048-sector boundaries
Total free space is 1282071 sectors (626.0 MiB)

Number  Start (sector)  End (sector)  Size     Code  Name
1       2048            2099199       1024.0 MiB  8300  Linux
filesystem
2       2623488         3147775       256.0 MiB   8300  Linux
filesystem
```

Особенности GPT-разметки:

– каждому диску присваивается уникальный идентификатор (GUID) — 128-битное шестнадцатеричное число. Благодаря этому при монтировании разделов используется GUID, а не путь устройства (например, /dev/sdb).

– GPT-диски поддерживают до 128 разделов, устраняя необходимость в первичных и расширенных разделах.

– свободное место отображается автоматически, поэтому отсутствует необходимость в команде `F`, как в fdisk.

3.7.2.2 Создание и удаление разделов

Создание нового раздела выполняется командой `n`.

Во время создания указываются номер раздела, начальный и конечный сектора (или размер), а также тип. GPT поддерживает больше типов разделов, чем MBR.

Просмотреть список доступных типов можно с помощью команды `l`.

Пример:

```
    Command (? for help): n
    Partition number (1-128): 3
    First sector (34-3903544, default = 34):
    Last sector, +sectors or +size{K,M,G,T,P} (default =
3903544): +512M
    Hex code or GUID (L to show codes, Enter = 8300):
```

Удаление раздела выполняется командой `d` и указанием номера необходимого раздела.

Если требуется переупорядочить нумерацию разделов без изменения их содержимого, используется команда `s`.

После выполнения этой команды таблица разделов будет перенумерована последовательно.

Следует учитывать, что при создании нового раздела можно использовать всё нераспределённое пространство диска, независимо от расположения существующих разделов.

3.7.2.3 Восстановление и резервирование данных GPT

GPT-диски содержат резервные копии заголовка и таблицы разделов, что облегчает восстановление в случае повреждения. Для доступа к функциям восстановления в gdisk используется команда `r (recovery mode)`.

Доступные операции восстановления:

– b и c – восстановление основного заголовка GPT или таблицы разделов;
– d и e – восстановление из резервной копии;
– f и g – преобразование схем разметки (из MBR в GPT и наоборот).

Для получения справки по доступным командам восстановления необходимо ввести знак "?" в режиме recovery.

3.7.2.4 Работа с дисками

При необходимости увеличить объём хранилища выполняется подключение и монтирование дополнительного диска на физическую или виртуальную машину. Для этого следует придерживаться алгоритма действий:

а) Подготовка. Необходимо проверить доступные в ОС накопители командой:

```
sudo fdisk -l
```

Иногда наличие устройства в BIOS может быть недостаточным.

б) Разметка диска через утилиту fdisk с указанием пути до диска:

```
fdisk /dev/sdb
```

Так как предполагается создавать простой раздел (не загрузочный) и используется все дисковое пространство, следует нажать клавиши `n` и `Enter`.

Система запросит тип раздела выбрать p(primary) – первичный.

Номер раздела – `1`.

На вопрос о первом и последнем секторе нажать клавишу `Enter`.

Следует обратить внимание, что если требуется создать несколько логических дисков, то в ответе на последний сектор следует указать размер в kilo-, mega-, giga-, tera-, petabytes. Соответственно, если раздел нужен размером в 2 гигабайта, то нужно указать 2Гб.

Далее нужно сохранить изменения нажатием клавиши `w` и подтвердить выбор с помощью клавиши `Enter`.

После выполнения операции, описанной выше, в Системе будет создано устройство /dev/sdb1.По своей сути это и есть раздел на диске.

в) Форматирование раздела. ОС предлагает на выбор несколько вариантов ФС. Создание ФС происходит выполнением команды `mkfs` с указанием ключей либо выполнением одной из программ:

```
mkfs.bfs
mkfs.btrfs
mkfs.cramfs
mkfs.ext2
mkfs.ext3
mkfs.ext4
mkfs.ext4dev
mkfs.fat
mkfs.minix
```

```
mkfs.msdos
mkfs.ntfs
mkfs.vfat
mkfs.xfs
```

Форматирование выполняется командой:

```
sudo mkfs.ext4 /dev/sdb1
```

Диск готов к работе. Остается только смонтировать его.

г) Монтирование. В ОС существует возможность монтировать диск (либо иное блочное устройство) в любую из директорий, которая должна быть пустой. Для этого нужно выполнить следующие действия:

1) создать каталог в директории /mnt:

```
sudo mkdir /mnt/1
```

2) изменить права доступа к каталогу (всем разрешаем всё):

```
sudo chmod -R 0777 /mnt/1
```

3) монтировать:

```
sudo mount /dev/sdb1 /mnt/1
```

Для монтирования диска автоматически при загрузке Системы необходимо редактировать файл /etc/fstab с помощью любого текстового редактора, например, nano:

```
sudo nano /etc/fstab
```

Далее в самый конец файла вставить строку и сохранить файл:

```
/dev/sdb1 /mnt/1 ext4 defaults 0 0
```

3.7.2.5 Образы дисков

Для создания и записи образов дисков в Системе используются утилиты dd, genisoimage и wodim, обеспечивающие полное управление процессом копирования и подготовки носителей.

Для записи информации на новый диск предварительно создаётся его образ.

Для **создания копии существующего диска** необходимо выполнить:

```
dd if=/dev/cdrom of=mycd.iso
```

где

– if — источник (input file);

– of — выходной файл (output file).

Утилита dd копирует содержимое устройства поблочно, создавая точный образ.

При необходимости можно задать размер блока (bs=1024) или продолжить копирование при ошибках чтения (noerror).

Примечание — утилита dd не подходит для копирования защищённых или двухслойных дисков.

Для **создания образа из каталога необходимо выполнить**:

```
genisoimage -f -v -J -o mycd.iso /mnt/somedisk/temp
```

Параметры:

– -v – вывод хода выполнения;
– -J – поддержка длинных имён файлов (Joliet);
– -f – следование по символическим ссылкам.

Для подготовки содержимого можно создать временный каталог с символическими ссылками на нужные файлы и передать его утилите genisoimage.

При создании DVD-видео необходимо включить файловую систему UDF, используя параметр -udf или -dvd-video.

Расширение Rock Ridge (-r) обеспечивает поддержку атрибутов UNIX и увеличенной глубины каталогов.

3.7.2.5.1 Запись образа на диск

Для записи подготовленного ISO-образа применяется утилита wodim:

```
wodim dev=/dev/hdc -eject -v mycd.iso
```

Параметры утилиты:

– -eject – извлечение диска после записи;
– -dummy – пробная запись без лазера;
– -multi – многосессионный диск;
– blank=fast или blank=all – очистка перезаписываемого носителя;
– speed=<значение> – установка скорости записи;
– -overburn – запись сверх стандартного объёма.

3.7.3 Диспетчер разделов KDE

Диспетчер разделов KDE — стандартный графический инструмент для управления разделами в среде KDE. Он предоставляет удобный интерфейс для выполнения операций над разделами, включая создание, удаление, изменение

размеров и перемещение. Диспетчер разделов поддерживает все основные типы файловых систем, используемых в Unix-подобных системах.

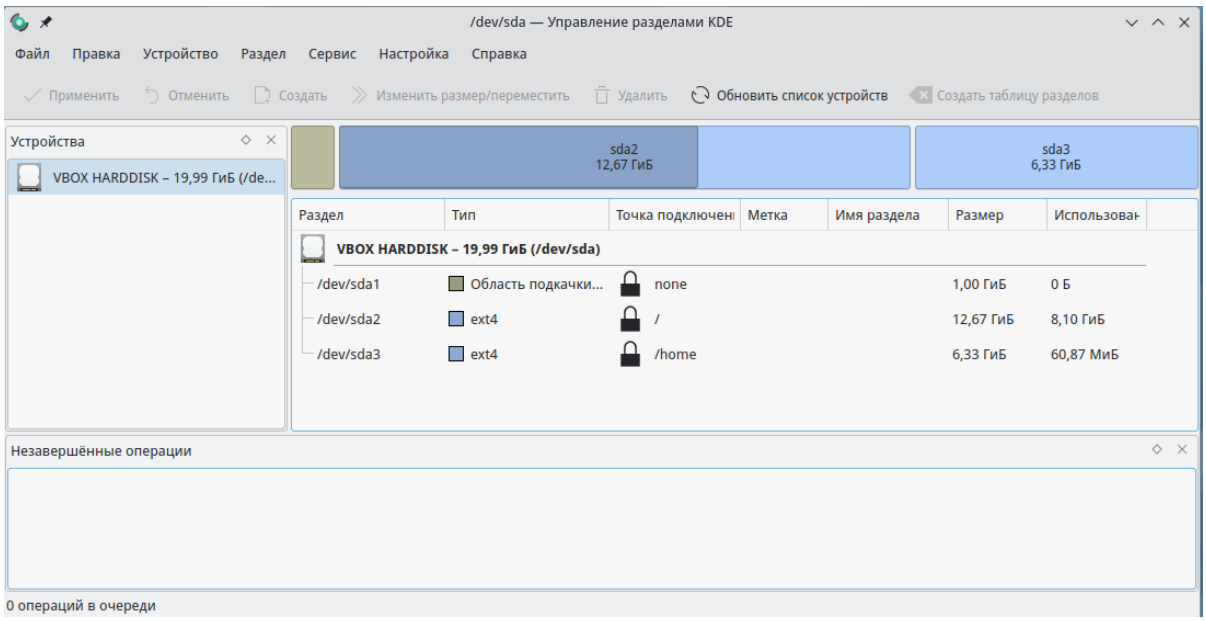


Рисунок 77 – Диспетчер разделов KDE

Для перехода к приложению нужно войти в "Главное меню Системы → Утилиты → Диспетчер разделов KDE"(рисунок 77).

Диспетчер разделов KDE предоставляет следующие основные функции:

– просмотр информации о дисках и разделах: о размере, файловой системе и свободном месте;

– создание новых разделов с выбором файловой системы;

– удаление разделов;

– изменение размеров разделов без потери данных, если файловая система поддерживает такие операции;

– форматирование разделов в различные файловые системы, такие как ext4, xfs, btrfs и другие;

– присвоение меток разделам для удобства их идентификации.

Практические инструкции по работе с Диспетчером разделов KDE:

а) Для создания нового раздела нужно:

1) выбрать диск, на котором будет создан новый раздел;

2) в верхнем навигационном меню программы перейти в "Раздел → Создать";

3) задать тип раздела (например, основной или логический) и размер раздела;

4) выбрать файловую систему для нового раздела, например, ext4;

5) при необходимости задать метку раздела;

б) нажать на кнопку Применить, чтобы подтвердить создание нового раздела. Изменения будут применены только после подтверждения.

Для изменения размера раздела нужно:

1) выбрать раздел, который необходимо изменить;

2) по нажатию правой кнопкой мыши на разделе выбрать кнопку Изменить размер;

3) в открывшимся окне ввести новый желаемый размер раздела (рисунок 78);

4) нажатием на кнопку ОК сохранить все внесенные изменения;

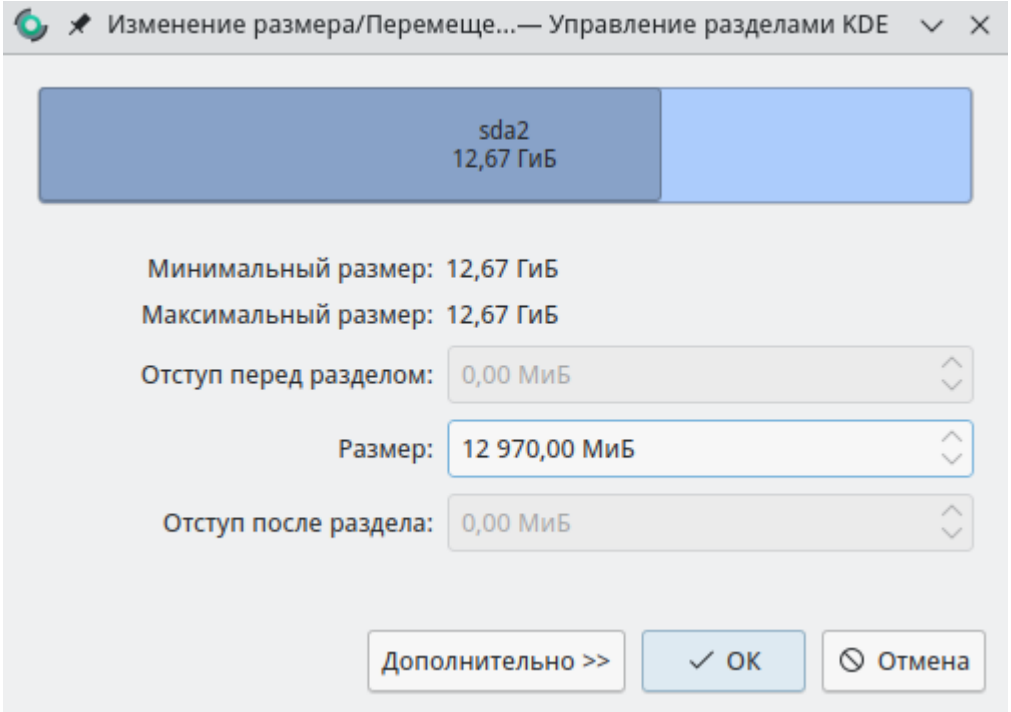


Рисунок 78 – Изменение размера раздела в Диспетчере разделов KDE

в) Для удаления раздела нужно:

1) выбрать раздел, который необходимо удалить;

2) по нажатию правой кнопкой мыши на разделе в контекстном меню выбрать кнопку Удалить;

3) для подтверждения действия нажать на кнопку Применить.

Важно – Перед выполнением операций с разделами рекомендуется создать резервные копии важных данных, так как изменение разделов может привести к потере данных.

Важно – Следует быть осторожными при работе с разделами, на которых установлена ОС или важные данные, чтобы избежать потери доступа к Системе.

3.7.4 GParted

GParted — это мощный и гибкий графический инструмент для управления разделами жесткого диска. Он поддерживает широкий спектр файловых систем и позволяет выполнять разнообразные операции с дисками и разделами, такие как создание, удаление, изменение размера и перемещение разделов. GParted особенно полезен для сложных задач по переразметке дисков и работает в среде GNOME.

GParted позволяет просматривать и отменять действия до их применения. Также можно видеть все запланированные изменения в журнале перед их окончательной реализацией.

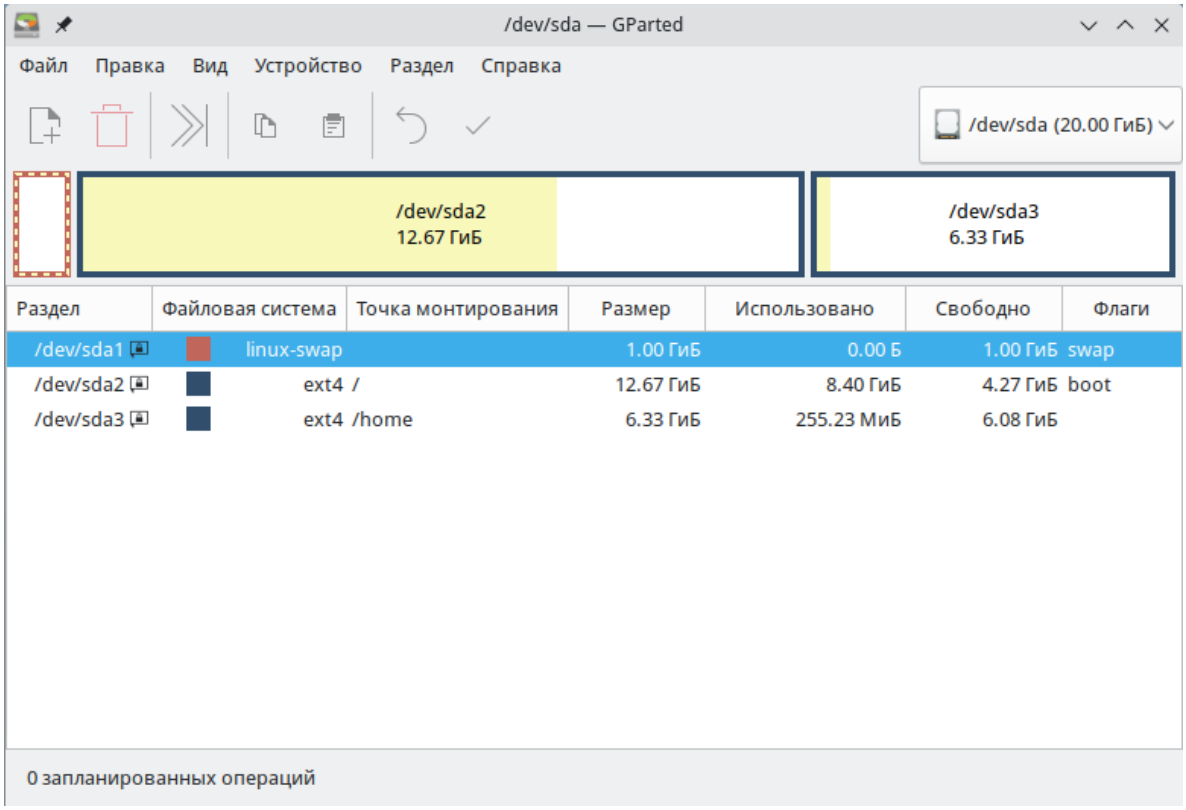


Рисунок 79 – Приложение GParted

Для перехода к приложению нужно войти в "Главное меню Системы → Все приложения → GParted". После запуска GParted открывается основной интерфейс программы (рисунок 79):

– Меню выбора диска: в верхнем правом углу в выпадающем списке можно выбрать нужный диск, если у вас несколько физических дисков.

– Основное окно, в котором отображаются все разделы выбранного диска в виде графической схемы.

– Информация о разделе: под схемой отображается информация о выделенном разделе: его файловая система, размер, статус (занятое и свободное место), а также метка.

Основные операции с разделами:

а) Для создания нового раздела нужно:

1) выбрать диск с неразмеченным пространством;

2) по нажатию правой кнопкой мыши по неразмеченному пространству на диске в открывшимся контекстном меню выбрать кнопку Создать (пиктограмма );

3) в открывшемся окне выбрать размер раздела, файловую систему (например, ext4, NTFS и т.д.) и метку;

4) нажать на кнопку Добавить, чтобы сохранить изменения;

5) для применения изменений нажать на кнопку Применить (пиктограмма  в верхней панели инструментов).

б) Для изменения размера раздела нужно:

1) выбрать раздел, который необходимо изменить;

2) нажать на кнопку Изменить размер или перенести выбранный раздел (пиктограмма );

3) ввести новый размер раздела;

4) нажать на кнопку Применить (пиктограмма ), чтобы сохранить изменения.

в) Для удаления раздела нужно:

1) выбрать раздел, который необходимо удалить;

2) нажать на кнопку Удалить выбранный раздел (пиктограмма );

3) подтвердить действие нажатием на кнопку Применить.

а) Для форматирования раздела нужно:

1) выбрать раздел, который нужно отформатировать;

2) нажатием правой кнопкой мыши по разделу выбрать опцию "Форматировать в..." или перейти в пункт навигационного меню "Раздел → Форматировать в...", затем выбрать файловую систему, в которую необходимо отформатировать раздел;

3) подтвердить форматирование нажатием на кнопку Применить;

б) Для создания таблицы разделов:

1) выбрать раздел, с которым будет продолжена работа;

2) перейти в навигационном меню в раздел "Устройство → Создать таблицу разделов";

3) в открывшемся окне при необходимости указать дополнительные параметры.

Важно – Перед выполнением операций с разделами рекомендуется создать резервные копии важных данных, так как изменение разделов может привести к потере данных.

3.7.5 Управление дисковыми квотами

Настройка дисковых квот позволяет ограничивать объем дискового пространства, доступный пользователям и группам, а также контролировать потребление ресурсов на уровне файловой системы. Поддержка квот реализована средствами подсистемы diskquota.

Механизм квотирования обеспечивает:

– ограничение дискового пространства (block quota);
– ограничение количества создаваемых inode (inode quota);
– раздельное применение ограничений для пользователей и групп;
– регистрацию событий превышения лимитов в системных журналах;
– управление превышениями через мягкие и жёсткие лимиты;
– автоматическое уведомление пользователя при достижении мягкого лимита.

Поддержка квот актуальна для файловых систем ext4, xfs и других, входящих в состав ОС. Настройка выполняется отдельно для каждого раздела файловой системы.

Перед включением квот необходимо проверить поддержку соответствующих опций монтирования. Для файловой системы ext4 требуется указать параметры usrquota и/или grpquota.

Для постоянного включения квот следует отредактировать запись в файле /etc/fstab. Пример строки:

```
/dev/sda1   /home   ext4   defaults,usrquota,grpquota   0 2
```

После внесения изменений файловая система должна быть перемонтирована:

```
mount -o remount /home
```

Для файловой системы xfs параметры квот задаются при монтировании:

```
mount -o uquota,gquota /home
```

После включения опций квотирования требуется создать или обновить служебные файлы квот. Для ext4 используется команда:

```
quotacheck -cug /home
```

где:

– -u — проверка пользовательских квот;
– -g — проверка групповых квот;
– -c — создание новых файлов квот.

Для ускорения проверки рекомендуется предварительно временно отключить квоты:

```
quotaoff /home
quotacheck -avug
```

После завершения проверки квоты включаются:

```
quotaon /home
```

Статус квот можно просмотреть командой:

```
quotaon -p /home
```

Для назначения пользовательских и групповых квот используется утилита edquota. Редактирование квот для пользователя:

```
edquota имя_пользователя
```

На экране редактора указываются значения:

– Soft limit — мягкое ограничение: задаёт порог допустимого использования дискового пространства или числа файлов, при превышении которого для пользователя начинается отсчёт льготного периода (grace period), однако работа с файловой системой остаётся доступной;

– Hard limit — жёсткое ограничение: представляет собой абсолютный предел, превышение которого невозможно и при достижении которого дальнейшая запись данных или создание файлов блокируется немедленно;

– Inode soft/hard — ограничения на количество файлов: регулируют не объём данных, а максимально допустимое количество файлов и каталогов, которые может создать пользователь или группа.

Период превышения мягкого лимита определяет время, в течение которого пользователь может превышать soft limit без блокировки операций. Значение задаётся командой:

```
edquota -t
```

Указывается время для блоков и inode, например:

– 7days;
– 12hours.

По истечении срока превышения пользователь не сможет создавать файлы или занимать новое пространство до освобождения ресурса или увеличения лимита.

Аналогично выполняется настройка квот для группы:

```
edquota -g имя_группы
```

Для копирования квот одного пользователя другим используется:

```
edquota -p пользователь_образец пользователь1 пользователь2
```

Для просмотра текущего состояния квот применяется команда:

```
quota -u имя_пользователя
```

Для просмотра групповых квот:

```
quota -g имя_группы
```

Для обзора квот по всем пользователям используется repquota:

```
repquota -a
```

Отчёт формируется по каждому смонтированному разделу с поддержкой квот, включая сведения о фактическом использовании блоков и inode.

События, связанные с превышением квот, фиксируются подсистемой journald и могут быть просмотрены с помощью:

```
journalctl -u quota
```

Также возможен фильтр по ключевому слову:

```
journalctl | grep quota
```

3.7.6 Блокирование файлов (fileprotect)

Пакет fileprotect обеспечивает возможность блокирования изменений определённых файлов и каталогов независимо от прав доступа, заданных в файловой системе. Механизм предназначен для защиты критически важных конфигурационных файлов от модификации, удаления и перезаписи как пользователями, так и прикладными процессами.

Механизм fileprotect позволяет:

– помечать файлы и каталоги как защищённые от изменений;
– запрещать операции записи, переименования и удаления;

– вести журнал событий обращения к защищённым объектам;

– применять настройки как на уровне отдельных файлов, так и целых каталогов.

Ограничения применяются средствами подсистемы ядра, что обеспечивает невозможность обхода защиты через обычные механизмы изменения прав.

Настройка осуществляется с помощью утилиты fileprotectctl. Для добавления файла в список защищённых используется команда:

```
fileprotectctl add /путь/к/файлу
```

Для удаления файла из защиты:

```
fileprotectctl remove /путь/к/файлу
```

Для просмотра текущего списка защищённых объектов:

```
fileprotectctl list
```

При необходимости может быть заблокирован целый каталог. В этом случае защита применяется ко всем вложенным объектам.

3.7.6.1 Журналирование событий

Сведения о попытках изменения защищённых файлов фиксируются в системном журнале и могут быть просмотрены с помощью команд:

```
journalctl -u fileprotect
```

или с фильтром по ключевому слову:

```
journalctl | grep fileprotect
```

Журналирование позволяет выявлять процессы или пользователей, предпринимающих попытки модификации критически важных файлов.

При временной необходимости модификации конфигурации защита снимается:

```
fileprotectctl disable /путь/к/файлу
```

После завершения изменения объект может быть снова защищён:

```
fileprotectctl enable /путь/к/файлу
```

Удаление пакета fileprotect не рекомендуется, поскольку это приводит к потере ранее назначенных правил и лишает Систему дополнительного уровня защиты файловой структуры.

3.8 Режимы работы Системы

Функционирование ОС возможно в трех режимах – нормальном, в режиме восстановления и в аварийном режиме. По умолчанию при включении ПК ОС загружается в нормальном рабочем режиме, для включения других режимов необходимо добавление параметров загрузки в меню начального загрузчика.

Для работы в режиме восстановления и в аварийном режиме в общем случае нужны полномочия администратора Системы.

Для изменения параметров загрузчика нужно в начальном меню GRUB нажать клавишу `E`. В открывшемся редакторе вписать параметр загрузки в строке, начинающейся с "linux /boot/vmlinuz...". Например, рядом с параметром `quet`. После этого нужно нажать клавишу `F10` для загрузки ОС с указанными параметрами.

И режим восстановления, и аварийный режим являются однопользовательскими; в них невозможна одновременная авторизация нескольких пользователей, также в них не активируются сетевые интерфейсы.

3.8.1 Режим восстановления

В ОС есть возможность загрузки в режиме восстановления (rescue mode) без графической оболочки и необязательных сервисов. Этот режим дает шанс восстановить Систему, которая перестала загружаться в графическом режиме, например, при сбое обновления или при указании нерабочих настроек в файлах конфигурации.

Для загрузки ОС в режиме восстановления необходимо при включении Системы, как описано в п.3.8, загрузиться с параметром systemd.unit=rescue.target.

После такой загрузки возможен вход в Систему с помощью консольного интерфейса. Система попытается смонтировать локальные разделы обычным образом, и появится возможность изменять файлы конфигурации редактором nano или встроенным редактором текстовой оболочки mc.

3.8.2 Аварийный режим

Аварийный режим (emergency mode) используется для восстановления ОС или спасения данных в случаях серьезных ошибок функционирования, например, при повреждениях ФС, когда не удается войти даже в режиме восстановления.

Для загрузки Системы в аварийном режиме необходимо при включении Системы, как описано в п.3.8, загрузиться с параметром systemd.unit=emergency.target.

После такой загрузки возможен вход и управление ОС с помощью консольного текстового интерфейса. В аварийном режиме корневой раздел монтируется "только для чтения", другие разделы не монтируются, и сетевые интерфейсы не активируются.

3.8.3 Синхронизация с репозиториями

Если сбой ОС произошел во время обновления вследствие проблем с сетью или ошибок питания, есть возможность в консольном интерфейсе запустить сеть командой:

```
sudo systemctl start NetworkManager
```

При необходимости установить сетевое соединение командой `nmtui` и синхронизировать текущую Систему с репозиториями, приведя версии ее пакетов в соответствие с актуальными. Это можно сделать командой:

```
sudo dnf distrosync
```

Следует обратить внимание, что при такой операции могут быть удалены сторонние пакеты, установленные не из подключенных репозиториев.

3.8.4 Восстановление функционирования ОС

3.8.4.1 Переустановка ОС с сохранением данных

Самым быстрым и простым способом восстановления Системы является ее переустановка с инсталляционного образа с сохранением пользовательских данных. В процессе установки важно не форматировать раздел /home, где хранятся пользовательские данные. Для этого в инсталляторе ОС РОСА "Хром" необходимо вручную выбрать раздел /home, убедившись, что не установлен флажок "Форматировать". Таким образом, все пользовательские файлы и настройки, находящиеся в этом разделе, сохранятся, а Система будет переустановлена в оптимальном состоянии.

Это подходит в случаях, когда системные файлы повреждены, но данные пользователей остаются целыми.

3.8.4.2 Использование утилиты TestDisk

В состав ОС включена утилита TestDisk, которая предназначена для восстановления потерянных данных и восстановления разделов, если произошли ошибки файловой системы. TestDisk может быть использован в ситуациях, когда произошел сбой в работе дисков или были случайно удалены разделы. Утилита поддерживает восстановление различных файловых систем.

Для восстановления данных с помощью TestDisk нужно:

а) Установить TestDisk (если утилита не была предустановлена) командой:

```
sudo dnf install testdisk
```

Запустить TestDisk с правами суперпользователя:

```
sudo testdisk
```

Выбрать диск, с которым возникли проблемы. Далее следовать пошаговым инструкциям на экране. Утилита проанализирует диск и предложит возможные способы восстановления разделов или данных.

3.8.4.3 Замена носителя и восстановление данных из резервной копии

В случае, когда носитель полностью вышел из строя и восстановление данных с помощью TestDisk невозможно, рекомендуется заменить неисправный диск на новый. После этого необходимо восстановить данные из созданных ранее резервных копий. Для этого подойдет любая созданная ранее резервная копия (например, с использованием инструментов резервного копирования, таких как rsync, Bacula или Tar). Подробнее о резервном копировании данных см. раздел 7.2.

Ключевые шаги при замене носителя и восстановлении данных:

– замена физического носителя (жесткого диска или SSD);
– загрузка с инсталляционного образа и установка ОС;
– восстановление данных из резервной копии после установки копированием их на новый носитель.

Восстановление из резервной копии позволяет минимизировать потери данных и ускоряет процесс восстановления Системы в случае серьезных сбоев.

3.8.5 Восстановление загрузчика GRUB2 (MBR и EFI)

В случае повреждения загрузочной записи GRUB2 может потребоваться её восстановление вручную. Для выполнения восстановления рекомендуется использовать Live-образ ОС РОСА или совместимого дистрибутива Linux с доступом к сети и средствам управления разделами (например, GParted).

Порядок восстановления загрузчика:

а) загрузка с Live-носителя (USB или DVD) с возможностью запуска в Live-режиме (без установки ОС);

б) определение корневого раздела и загрузочного диска ОС. Для этого рекомендуется воспользоваться утилитой fdisk или графическим интерфейсом

GParted, доступным в Live-среде. Утилита fdisk выводит список разделов с помощью команды:

```
fdisk -l
```

В результате будет выведен список всех доступных дисков и разделов. Далее требуется определить:

– корневой раздел, на котором установлена ОС РОСА (обычно это раздел типа Linux, например, /dev/sda2);

– загрузочный диск, на который необходимо установить загрузчик GRUB2 (обычно это физический диск, например, /dev/sda).

Полученные значения будут использоваться в дальнейших командах по монтированию и установке загрузчика:

а) Монтирование Системы:

```
sudo -i
mkdir /chroot
sudo mount /dev/sdXX /chroot
```

где sdXX необходимо заменить на имя корневого раздела.

При использовании файловой системы BTRFS:

```
sudo mount -o subvol=root /dev/sdXX /chroot
```

где sdXX необходимо заменить на имя корневого раздела.

б) Монтирование необходимых виртуальных файловых систем и конфигураций:

```
for i in dev proc sys etc/resolv.conf; do mount --bind /$i\
/chroot/$i; done
```

Если используется EFI-загрузка, необходимо также смонтировать переменные прошивки:

```
sudo mount --bind /sys/firmware/efi/efivars\
/chroot/sys/firmware/efi/efivars
```

в) Переход в chroot-окружение командой:

```
chroot /chroot
```

г) Монтирование всех разделов из fstab:

```
sudo mount -a
```

Это обеспечит доступ к дополнительным разделам, включая /boot и /boot/efi, если таковые имеются.

д) Установка загрузчика GRUB2 в главный загрузочный сектор:

```
sudo grub2-install /dev/sdX
```

где sdX необходимо заменить на имя загрузочного диска.

е) Обновление конфигурации загрузчика. После установки необходимо сгенерировать конфигурационный файл GRUB2:

```
sudo update-grub2
```

Это обеспечит добавление всех обнаруженных систем (например, Windows) в загрузочное меню.

ж) Выход из окружения chroot и перезагрузка ОС:

```
exit
sync
reboot
```

Также есть альтернативный способ восстановления загрузчика GRUB2: можно воспользоваться утилитой grub2recovery, доступной в репозитории contrib. Для её установки нужно выполнить команду:

```
sudo dnf install grub2recovery
```

Подробности о применении утилиты доступны на официальном форуме: https://forum.rosa.ru/viewtopic.php?f=58&t=10649.

3.8.6 Восстановление Системы из среды chroot

В случае повреждения Системы или необходимости выполнения административных операций без загрузки основной ОС может использоваться метод восстановления через среду Live (LiveCD/LiveUSB) с переходом в установленную Систему в режиме chroot. Для этого нужно выполнить следующие шаги:

а) Загрузить в Live-среду с установочного носителя (например, USB или DVD) в Live-режиме, выбрав пункт "Запустить без установки" или аналогичный.

б) Определить корневой раздел и загрузочный диска ОС. Для этого рекомендуется воспользоваться утилитой fdisk или графическим интерфейсом GParted, доступным в Live-среде. В терминале нужно выполнить команду:

```
fdisk -l
```

В результате будет выведен список всех доступных дисков и разделов. Требуется определить:

– корневой раздел, на котором установлена ОС РОСА (обычно это раздел типа Linux, например, /dev/sda2);

– загрузочный диск, на который необходимо установить загрузчик GRUB2 (обычно это физический диск, например, /dev/sda).

Полученные значения будут использоваться в дальнейших командах по монтированию и установке загрузчика.

смонтировать корневую файловую систему в каталог, подготовленный для chroot (во всех командах параметр sdXX необходимо заменить на имя коревого раздела):

```
mkdir /chroot
sudo mount /dev/sdXX /chroot
```

Если используется файловая система BTRFS, указать нужный подтом:

```
sudo mount -o subvol=@ /dev/sdXX /chroot
```

Подключить необходимые виртуальные файловые системы:

```
for i in dev sys proc; do mount --bind -v /$i /chroot/$i;\
done
```

Для обеспечения работы сетевых функций (например, получения пакетов или использования утилит) скопировать файл конфигурации DNS:

```
cp /etc/resolv.conf /chroot/etc/resolv.conf
```

а) Войти в установленную Систему с помощью команды:

```
chroot /chroot
```

После выполнения данной команды будет организован доступ к установленной Системе в режиме полного административного управления.

б) Выполнить все необходимые операции в среде chroot, включая:

– изменение паролей;
– переустановка загрузчика;
– исправление конфигурационных файлов;
– установка или удаление пакетов.

в) Завершить работу и перезагрузить Систему. После завершения всех операций необходимо выйти из chroot по команде:

```
exit
```

Отключить файловые системы (при необходимости) и перезагрузить Систему командой:

```
reboot
```

После перезагрузки будет выполнена загрузка основной установленной Системы.

3.9 Перезагрузка и остановка Системы

Использование различных способов перезагрузки и остановки Системы позволяет гибко управлять её состоянием в зависимости от текущих задач. Выбор конкретного метода зависит от ситуации и уровня доступа пользователя.

Перезагрузка Системы может потребоваться для:

– применения обновлений Системы;

– изменения в конфигурационных файлах, которые применяются только после перезагрузки;

– устранения сбоев в Системе.

3.9.1 Перезагрузка

Для выполнения перезагрузки через терминал можно воспользоваться одним из следующих способов:

– использовать команду, которая немедленно инициирует перезагрузку Системы:

```
sudo reboot
```

В случае зависания Системы можно использовать принудительную перезагрузку:

```
sudo reboot -f
```

Опция `-f` (`--force`) перезагружает Систему без учёта активных процессов и может использоваться в крайних случаях.

– перезагрузить Систему с использованием команды `init`, которая позволяет указать конкретный уровень выполнения (для перезагрузки используется уровень 6):

```
sudo init 6
```

– С использованием systemd, предпочтительно использовать следующую команду, которая обеспечивает корректную остановку сервисов перед перезагрузкой:

```
sudo systemctl reboot
```

– С использованием команды `shutdown` (подробное использование рассмотрено в разделе 3.9.3):

```
sudo shutdown
```

3.9.2 Остановка

Остановка Системы может выполняться для её полного выключения или планового технического обслуживания.

Для остановки Системы через командную строку можно использовать одну из следующих команд:

– Завершение работы всех сервисов и выключение ПК:

```
sudo poweroff
```

– Остановка с использованием команды `systemctl`:

```
sudo systemctl poweroff
```

– Принудительное выключение Системы можно выполнить с помощью команды:

```
sudo poweroff -f
```

Важно – использовать эту команду можно только при необходимости, так как она может привести к потере данных.

– Остановка Системы с помощью команды `shutdown`. Система завершит работу через 1 минуту:

```
sudo shutdown
```

При подключении по SSH используются те же команды, что и в локальном терминале. Например:

```
ssh user@hostname sudo reboot
```

В случае если Система полностью зависла и не реагирует на команды, можно использовать аппаратные кнопки на корпусе:

– Кнопка перезагрузки (Reset): перезагружает Систему принудительно.

– Кнопка включения/выключения (Power): нужно удерживать кнопку несколько секунд для полного выключения Системы.

3.9.3 Использование команды shutdown

Команда `shutdown` — один из самых удобных и гибких инструментов для управления выключением, перезагрузкой и остановкой Системы. В этом разделе мы рассмотрим основные варианты использования команды `shutdown` с примерами.

Общий синтаксис команды:

```
sudo shutdown [OPTIONS] [TIME] [MESSAGE]
```

где:

OPTIONS – Дополнительные параметры, указывающие тип действия (выключение, перезагрузка и т. д.).

TIME – Время, через которое произойдет действие. Это может быть указано как абсолютное время (в формате "часы:минуты", например, 14:30) или как относительное время (например, +5 для действия через 5 минут) или с аргументом now для немедленного выполнения действия.

MESSAGE – Сообщение, которое будет отправлено всем пользователям перед завершением работы.

Основные параметры команды `shutdown` и их использование:

– Немедленное выключение Системы (по умолчанию):

```
sudo shutdown now
```

– Для перезагрузки Системы используется опция -r:

```
sudo shutdown -r now
```

Также возможно указать время перезагрузки, например, перезагрузка через 5 минут:

```
sudo shutdown -r +5
```

Или в определённое время, например, в 22:00:

```
sudo shutdown -r 22:00
```

– Отмена запланированного завершения или перезагрузки Системы с помощью опции -c:

```
sudo shutdown -c
```

Также возможно отправить сообщение всем пользователям о том, что выключение отменено:

```
sudo shutdown -c "Перезагрузка отменена."
```

– Выключение Системы с остановкой всех процессов и питания с помощью опции -h:

```
sudo shutdown -h now
```

Или аналогичным способом указать время выключения питания. Например, через 10 минут:

```
sudo shutdown -h +10
```

– Перевод Системы в спящий режим (halt) без полного выключения питания (в зависимости от настроек Системы и оборудования):

```
sudo shutdown -H now
```

3.10 Обновление Системы

3.10.1 Общие сведения

Пакеты обновлённой версии Системы размещаются в официальном репозитории разработчика. Список доступных репозиториев можно найти в установленной ОС в каталоге /etc/yum.repos.d/.

Загрузка пакетов обновлений осуществляется только из официальных репозиториев разработчика при наличии подключения к сети Интернет.

При загрузке пакетов обновлений из официального репозитория требуется использование персонализированного ключа, ограничивающего возможность получения обновлений третьими лицами. Персонализированные ключи предоставляются потребителю при получении лицензии на использование Системы.

При первом запуске ОС ключ будет запрошен в графическом интерфейсе программы обновления Системы. Также ключ может быть указан позже:

– через терминал: путем размещения его в файле /etc/dnf/vars/token. Для этого можно воспользоваться, например, текстовым редактором nano, и открыть файл командой:

```
sudo nano /etc/dnf/vars/token
```

– в графическом режиме: по нажатию правой кнопкой мыши на значок светофора в системном лотке и выбору в контекстном меню "Установить ключ репозитория".

3.10.2 Индикация обновлений

Система имеет наглядный графический инструмент для отображения информации об обновлениях – значок-индикатор светофор  в системном лотке:

– зелёный цвет – обновления отсутствуют;
– жёлтый цвет – найдены пакеты, требующие обновления;
– красный цвет – невозможно проверить наличие обновлений.

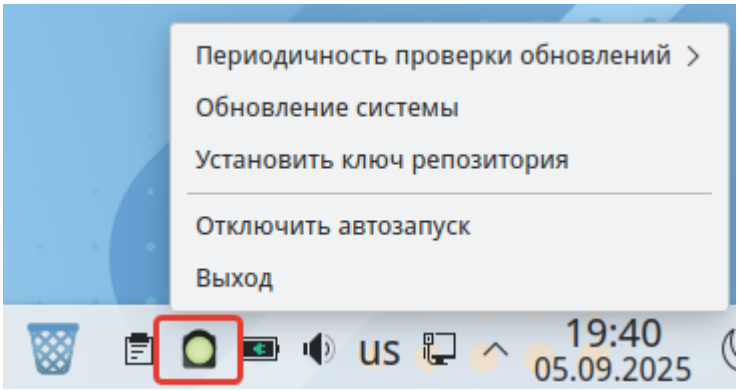


Рисунок 80 – Индикатор обновлений (светофор) и его контекстное меню

Для принудительной проверки обновлений Системы необходимо нажать правой кнопкой мыши на иконку светофора и в открытом контекстном меню (рисунок 80) выбрать "Обновление системы", после чего будет инициализирован процесс проверки обновлений пакетов. По завершении проверки выводится информационное окно о наличии или отсутствии пакетов с обновлениями.

Когда индикатор светофора горит желтым ( ) по наведению на него курсором мыши можно увидеть список пакетов, требующих обновления.

Для настройки периодичности проверки обновлений в ОС необходимо правой кнопкой мыши нажать на значок светофора, далее в контекстном меню выбрать пункт "Периодичность проверки обновлений" и необходимый параметр:

– 15 минут;
– 1 час;
– 6 часов;
– 12 часов;
– при старте Системы.

Система обновлений учитывает схему питания и заряд батареи ноутбука.

3.10.3 Порядок получения обновлений

Произвести обновление ОС возможно двумя путями:

– через терминал: необходимо запустить менеджер пакетов dnf и выполнить команду:

```
sudo dnf update
```

Данная команда проверяет наличие обновлений в репозиториях и устанавливает доступные пакеты.

– в графическом режиме: через контекстное меню индикатора обновлений (значок светофора) ⟶ "Обновление системы", после чего Система перейдет к проверке доступных пакетов и их установке. Прогресс будет отображаться в информационном окне (рисунок 81).

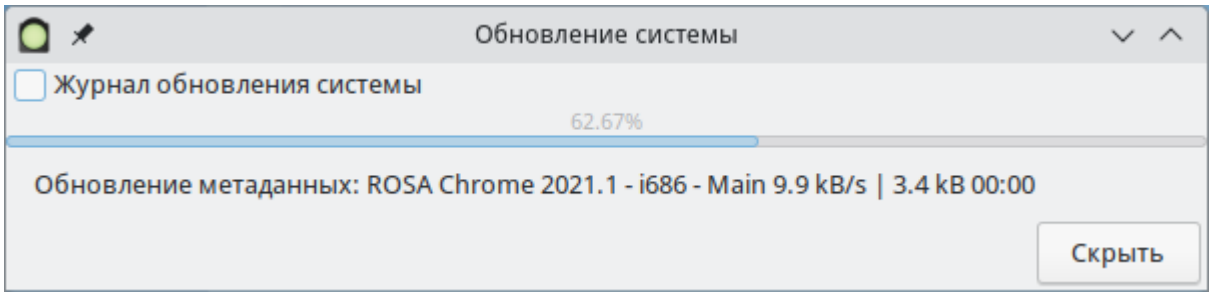


Рисунок 81 – Прогресс обновления Системы

3.11 Сохранение вывода консоли в файл

В процессе администрирования нередко возникает необходимость сохранить результаты выполнения команд терминала в файл – это может потребоваться для диагностики, документирования, пересмотра выполненных операций или подготовки отчётов.

Перенаправление вывода в файл

Для записи вывода команды в файл используется оператор перенаправления:

```
необходимая_команда parameters > logfile.txt
```

Если файл logfile.txt существует, его содержимое будет перезаписано. Для добавления новых данных к существующему содержимому следует использовать двойной оператор >>:

```
необходимая_команда parameters >> logfile.txt
```

Одновременный вывод на экран и в файл

Для одновременного отображения результата команды в терминале и записи в файл применяется утилита tee:

```
необходимая_команда parameters | tee -a logfile.txt
```

Опция -a указывает на добавление данных в конец файла без перезаписи.

Снимок экрана консоли с помощью setterm

Утилита setterm позволяет сохранить содержимое одной страницы виртуальной консоли:

```
setterm -dump -file /путь/к/файлу
```

Пример:

```
setterm -dump -file /root/screenlog
```

Содержимое текущей виртуальной консоли будет сохранено в указанный файл. Опции утилиты:

– -dump – создаёт дамп консоли;
– -file имя_файла – задаёт имя файла для сохранения;
– -append – добавляет данные к уже существующему файлу.

Для указания номера консоли используется форма:

```
setterm -dump [N] -file /root/screenlog
```

где N – номер консоли (например, 1 для /dev/tty1).

Использование утилиты script

Утилита script предназначена для записи всего, что происходит в терминале, включая команды и их вывод:

```
script --timing=terminal-timing.log terminal.log
```

После запуска запись начинается немедленно. Все действия будут сохранены в файл terminal.log, а временные характеристики – в terminal-timing.log.

Для завершения записи используется команда:

```
exit
```

Для воспроизведения записанного сеанса применяется одна из следующих команд:

```
scriptreplay terminal-timing.log terminal.log
```

или

```
scriptreplay terminal-timing.log terminal.log 2
```

где число в конце обозначает скорость воспроизведения (2 — в два раза быстрее и т.д.).

Копирование текста с помощью утилиты screen

Менеджер окон screen позволяет копировать данные из буфера прокрутки, для этого нужно:

а) запустить screen:

```
screen
```

б) перейти к выполнению нужных команд;

в) войти в режим копирования нажатием комбинации клавиш: `Ctrl+A`, затем `Ctrl+[`;

г) установить курсор на начало, нажать клавишу `Пробел`, затем переместить в конец и снова нажать клавишу `Пробел`;

д) переключиться на новое окно: использовать комбинацию клавиш `Ctrl+A`, затем нажать клавишу `c`;

е) перейти в новое окно: использовать комбинацию клавиш Ctrl+A, затем нажать клавишу 1;

ж) вставить текст: открыть редактор (например, mc), использовать две комбинации клавиш: сначала Ctrl+A, затем Ctrl+];

з) сохранить текст в редакторе;

и) вернуться в предыдущее окно: использовать комбинацию клавиш Ctrl+A, затем нажать клавишу 0.

Горячие клавиши в Konsole и Yakuake

В эмуляторах терминала Konsole и Yakuake предусмотрена встроенная функция сохранения вывода. По умолчанию используется сочетание клавиш `Ctrl+Shift+S`

Также можно изменить глубину буфера прокрутки в настройках. Стандартное значение — 1000 строк.

3.12 Запись экрана из консоли

При необходимости записи видеоролика с рабочего стола без привлечения графических программ может быть использована утилита ffmpeg, работающая в командной строке и поддерживающая большое количество аудио- и видеоформатов.

FFmpeg устанавливается из стандартного репозитория командой:

```
sudo dnf install ffmpeg
```

Для записи видео с экрана в формате QuickTime (MOV) без звука используется следующая команда:

```
ffmpeg -r 50 -g 500 -s 1366x768 -f x11grab -i :0.0 -vcodec\
qtrle screencast.mov
```

где параметры:

– -f x11grab – источник захвата, вывод X-сервера;
– -s 1366x768 – размер захватываемой области;
– -r 50 – частота кадров в секунду;
– -g 500 – интервал между ключевыми кадрами;

– -i :0.0 – экран X-сервера (дополнительно можно указать смещение: -i :0.0+10,20);

– -vcodec qtrle – видеокодек QuickTime RLE;

– screencast.mov – имя выходного файла.

Примечание – Параметры -r и -g подбираются эмпирически, в зависимости от производительности Системы и цели записи.

Для записи видео со звуком необходимо определить устройство захвата аудио, выполнив команду:

```
arecord -l
```

Пример использования FFmpeg с записью звука:

```
ffmpeg -f alsa -ac 1 -i hw:2,0 -acodec pcm_s16le -f x11grab\
-s 1366x768 -r 50 -g 500 -i :0.0 -vcodec qtrle screencast.mov
```

где параметры:

– -f alsa – захват звука через ALSA;

– -ac 1 – количество аудиоканалов (1 – моно, 2 – стерео);

– -i hw:2,0 – аудиоустройство (card 2, device 0);

– -acodec pcm_s16le – кодек без сжатия;

– остальные параметры аналогичны описанным в предыдущем примере.

3.12.1 Перекодирование видео

В случае необходимости сжатия или изменения формата видеозаписи возможно использование следующих команд:

– Простое перекодирование (по умолчанию):

```
ffmpeg -i screencast.mov screencast2.avi
```

– Кодирование с использованием libx264 в формат MKV:

```
ffmpeg -i screencast.mov -vcodec libx264 -crf 22 -threads 0\
-acodec libmp3lame -ar 44100 -ab 128k screencast2.mkv
```

– Кодирование с использованием libxvid в формат AVI:

```
ffmpeg -i screencast.mov -vcodec libxvid -qscale 8 -acodec\
libmp3lame -ab 128k -ac 2 screencast2.avi
```

где параметры:

– -crf – качество видео (чем меньше значение, тем выше качество);

– -qscale – аналогично crf, значение от 1 до 31 (меньше – лучше качество);

– -ac 2 – запись стерео;

– -threads 0 – использование всех доступных потоков CPU.

3.13 Процессы

Процесс – это программа, выполняемая в оперативной памяти компьютера. В многозадачной Системе одновременно могут быть активны множество процессов, каждый из которых использует выделенные ему ресурсы: процессорное время, память, устройства ввода-вывода.

Распределением ресурсов управляет планировщик ядра, который назначает приоритеты, переключает выполнение между процессами и контролирует использование памяти, исключая пересечения адресных пространств.

Все процессы в Системе создаются путём клонирования существующих с использованием системных вызовов clone(2) и fork(2). Новый (дочерний) процесс наследует окружение родительского и получает собственный идентификатор (PID).

3.13.1 Жизненный цикл процесса

Алгоритм работы процессов в ОС определяется следующими условиями:

– имеется запущенный процесс;
– каждый процесс может запускать подпроцессы;
– создание нового процесса создается клонированием исходного;
– прародителем всех процессов в системе является процесс init, запускаемый ядром ОС при загрузке;
– процессы взаимодействуют между собой посредством межпроцессного взаимодействия:

– каналы;
– сигналы;
– сокеты;
– разделяемая память;

– каждый процесс обладает свойствами (процесс обладает контекстом):

– PID – идентификатор процесса;
– PPID – идентификатор процесса, породившего данный процесс;
– UID и GID – идентификаторы прав процесса (соответствует UID и GID пользователя, от имени которого запущен процесс);
– приоритет процесса;
– состояние процесса (выполнение, сон и т.п.).

Типичный процесс в Linux-подобной ОС может быть представлен схемой (рисунок 82).

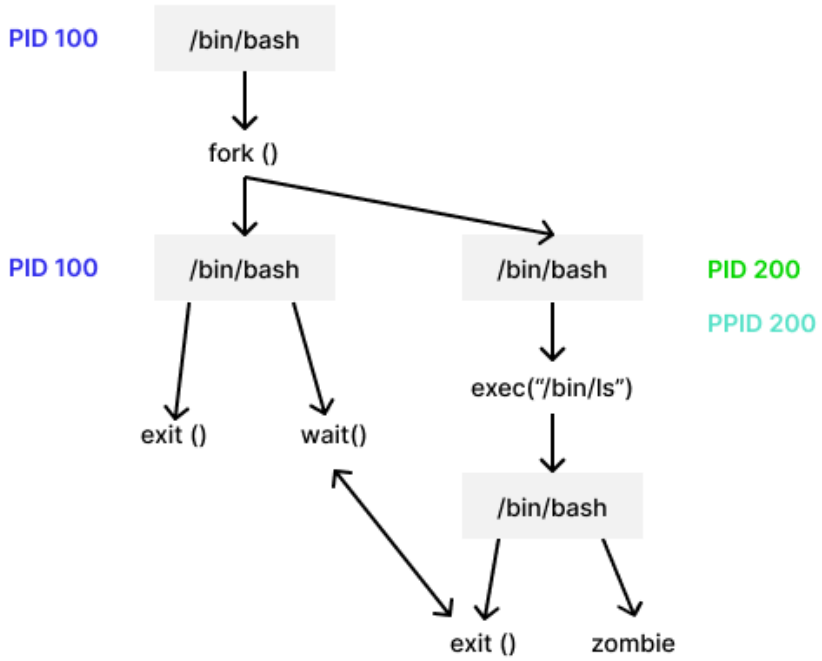


Рисунок 82 - Схема жизненного цикла процесса в ОС

Этапы состоят из следующих шагов:

а) процесс /bin/bash клонирует себя системным вызовом fork();

б) при этом создается клон /bin/bash с новым PID (идентификатор процесса) и PPID, равный PID родителя;

в) клон выполняет системный вызов exec с указанием на исполняемый файл и заменяет свой код кодом исполняемого файла (родительский процесс при этом ждет завершения потомка wait();

г) если по каким-то причинам потомок завершил свою работу, а родительский процесс не смог получить об этом сигнал, то данный процесс (потомок) не освобождает занятые структуры ядра и состояние процесса становиться zombie.

Первый процесс в Системе запускается при инициализации ядра. Данный процесс называется init и имеет PID=1 – это прародитель всех процессов в Системе.

Каждый запущенный процесс может находиться в одном из следующих состояний:

– R (Running) – активен, выполняется или ожидает выделения процессорного времени;

– S (Sleeping) – находится в прерываемом ожидании события или освобождения ресурса;

– D (Uninterruptible sleep) – ожидает сигнал от аппаратной части и не реагирует на другие сигналы;

– T (Stopped) – приостановлен, обычно в режиме трассировки при отладке;

– Z (Zombie) – завершён, но его структуры ядра не освобождены. Такое состояние возникает, если родительский процесс не получил сигнал о завершении дочернего;

– Сироты (Orphan) – процессы, потерявшие родителя. Они автоматически передаются процессу init, который обрабатывает сигналы об их завершении.

Процессы-зомби не используют ресурсы, но занимают записи в таблице процессов. Их появление свидетельствует о нарушении работы программ или оборудования и требует проверки состояния Системы.

Также, говоря о процессах в ОС, можно выделить особый вид процессов демоны. Данный вид процессов работает в фоне (подобно службам в Windows) без терминала и выполняет задачи для других процессов. Данный вид процессов на серверных системах является основным.

3.13.2 Приоритезация процессов

Каждому процессу при запуске назначается приоритет от −20 до +20, где:

– −20 – наивысший приоритет (процесс получает больше процессорного времени),

– +20 – низший приоритет.

Значение приоритета наследуется от родительского процесса.

Для изменения приоритета запускаемой программы используется утилита nice. Пример использования:

```
nice [-n adnice] команда [аргументы]
```

где adnice — числовое значение от −20 до +19, добавляемое к приоритету (значению nice) родительского процесса.

Отрицательные значения могут назначаться только суперпользователем.

Если параметр adnice не указан, приоритет процесса-потомка автоматически увеличивается на 10 по сравнению со значением родителя.

Изменить приоритет уже выполняющегося процесса позволяет команда `renice`. Обычные пользователи могут понижать приоритет только своих процессов, суперпользователь — любых.

Для управления процессами применяются сигналы. Наиболее часто используется команда `kill`:

```
kill [-SIG] PID
```

где SIG – это номер сигнала или наименование сигнала. Если сигнал не указан, по умолчанию отправляется SIGTERM (15) — корректное завершение процесса.

Для принудительного завершения любого процесса пользователем root применяется сигнал SIGKILL (9), приостанавливает выполнение — SIGSTOP (19), возобновляет — SIGCONT (18).

Обычные пользователи могут посылать сигналы только тем процессам, для которых они являются владельцами. Суперпользователь root может посылать сигналы любым процессам.

3.13.2.1 Работа в фоновом и переднем режиме

По умолчанию процессы выполняются на переднем плане. Для запуска в фоновом режиме используется символ "&" в формате:

```
команда &
```

Команды `fg` и `bg` переводят процессы между передним и фоновым режимом соответственно. Список активных заданий отображается командой `jobs`.

Чтобы процесс продолжал выполняться после выхода из оболочки, применяется утилита nohup:

```
nohup команда &
```

Запущенный таким образом процесс будет игнорировать посылаемые ему сигналы (не игнорируются только сигналы SIGHUP и SIGQUIT).

3.13.3 Получение информации о процессе

Для получения информации о каком-либо процессе можно использовать существующие утилиты ps и top для просмотра информации о процессах.

Чтобы получить список всех процессов, достаточно ввести команду:

```
# ps aux
```

Следует обратить внимание, что некоторые процессы указаны в квадратных скобках "[ ]" – это процессы, которые входят непосредственно в состав ядра и выполняют важные системные задачи, например, такие как управление буферным кешем [pdflush] и организацией свопинга [kswapd],

которые требуют очень ответственное отношение. Остальная часть процессов относится к пользовательским.

Информация, получаемая по каждому процессу:

– PID, PPID – идентификатор процесса и его родителя;
– %CPU – доля процессорного времени, выделенная процессу;
– %MEM – процент используемой оперативной памяти;
– VSZ – виртуальный размер процесса;
– TTY – управляющий терминал;
– STAT – статус процесса:
    – R – выполняется;
    – S – "спит";
    – Z – "зомби";
    – < – повышенный приоритет;
    – + – в интерактивном режиме;
– START – время запуска;
– TIME – время исполнения на процессоре.

Команда `ps` делает моментальный снимок процессов в текущий момент. В отличие от нее команда `top` динамически выводит состояние процессов и их активность в реальном режиме времени.

Пример вывода команды `top`:

```
     14:32:49 up 35 days, 6:01, 4 users, load average: 0.65, 0.51,
0.49
     Tasks: 432 total, 1 running, 431 sleeping, 0 stopped, 0 zombie
     CPU0  :  1.6%us,  3.6%sy,  0.0%ni,  85.3%id,  9.2%wa,  0.0%hi,
0.3%si, 0.0%st
     CPU1  :  0.9%us,  1.9%sy,  0.0%ni,  96.9%id,  0.0%wa,  0.0%hi,
0.3%si, 0.0%st
     Mem:  1033596K  total,  1016644K  used,  16952K  free,  82928K
buffers
     Swap: 2096376K  total,  12632K used,  2083744K  free,  478220K
cached
     PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
     2788 root 15 -5 0 0 0 S 2.0 0.0 404:43.97 md3_raid1
     7961 root 20 0 5964 2528 1684 R 2.0 0.2 0:00.14 top
     6629 root 20 0 8612 2884 2096 S 0.7 0.3 0:00.96 dovecot-auth
     57 root 15 -5 0 0 0 S 0.3 0.0 4:36.10 kblockd/0
     8703 ulogd 20 0 17700 4216 656 S 0.3 0.4 87:23.98 ulogd
```

```
11336 ldap 20 0 394M 15M 8292 S 0.3 1.5 5:29.28 slapd
25757 ldap 20 0 394M 15M 8292 S 0.3 1.5 5:11.71 slapd
10991 root 20 0 2188 1004 588 S 0.3 0.1 4:23.33 dovecot
1 root 20 0 1712 516 464 S 0.0 0.0 0:46.17 init
2 root 15 -5 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root RT -5 0 0 0 S 0.0 0.0 0:05.92 migration/0
.....
2960 root 16 -4 1980 520 392 S 0.0 0.1 0:10.04 udevd
2993 dovecot 20 0 4412 1800 1476 S 0.0 0.2 0:00.00 pop3-login
2994 dovecot 20 0 4412 1800 1476 S 0.0 0.2 0:00.02 pop3-login
```

В верхней части вывода отображается астрономическое время, время, прошедшее с момента запуска Системы, число пользователей в Системе, число запущенных процессов и число процессов, находящихся в разных состояниях, данные об использовании ЦПУ, памяти и свопа. Далее следует таблица, характеризующая отдельные процессы. Число строк, отображаемых в этой таблице, определяется размером окна: сколько строк помещается, столько и выводится.

Содержимое окна обновляется каждые 5 секунд. Список процессов может быть отсортирован по используемому времени ЦПУ (по умолчанию), по использованию памяти, по PID, по времени исполнения. Переключать режимы сортировки при отображении можно с помощью следующих клавиатурных команд:

– `Shift+N` – по PID;
– `Shift+A` – по возрасту;
– `Shift+P` – по использованию ЦПУ;
– `Shift+M` – по использованию памяти;
– `Shift+T` – по времени выполнения.

С помощью команды `<K>` можно завершить некоторый процесс (его PID будет запрошен), а с помощью команды `<R>` можно переопределить значение nice для некоторого процесса.

Полезную информацию о процессах также позволяет получить программа lsof, которая выдает список всех файлов, используемых сейчас процессами, включая каталоги, используемые каким-либо процессом в качестве текущего или корневого; разделяемые библиотеки, загруженные в память и т. д.

3.13.4 Управление процессами

3.13.4.1 Команда free

Команда `free` используется для получения подробного отчета об использовании памяти Системой, предоставляет информацию об общем объеме физической памяти и памяти подкачки, а также о свободной и используемой памяти.

Синтаксис команды следующий:

```
free [OPTIONS]
```

При использовании без какой-либо опции команда `free` будет отображать информацию о памяти и подкачке в килобайтах.

Обозначения столбцов при выводе команды:

– total – общий объем памяти, который может быть использован приложениями.

– used – используемая память, которая рассчитывается как: used = total-free-buffers- cache.

– free – свободная/неиспользуемая память.

– Shared – можно игнорировать, поскольку не имеет значения (только для обратной совместимости);

– buff / cache – объединенная память, используемая буферами ядра, а также страничным кешем и блоками. Эта память может быть освобождена в любое время, если это необходимо приложениям.

– available – оценка объема памяти, доступного для запуска новых приложений без подкачки.

Если требуется, чтобы буферы и кеш отображались в двух отдельных столбцах, используют параметр -w.

По умолчанию команда `free` показывает информацию о памяти в килобайтах. Чтобы просмотреть информацию в удобочитаемом формате (обычно в мегабайтах и гигабайтах), используют параметр -h:

```
free -h
```

Команда `free` также позволяет указать единицы измерения памяти. Допустимые варианты:

– b, --bytes – отображать вывод в байтах;

– -kilo – отображать вывод в килобайтах (1 Кб = 1000 байт);

– -mega – отображать вывод в мегабайтах;

– -giga – показать вывод в гигабайтах;

– -tera – показать вывод в терабайтах;

– k, --kibi – показать вывод в кибибайтах. (1 Киб = 1024 байт) (по умолчанию);

– m, --mebi – показать вывод в мебибайтах (1 МиБ = 2^20 байт);

– g, --gibi – показать вывод в гибибайтах (1 ГиБ=2^30 байт);

– -tebi – отображать вывод в тебибайтах (1 ТиБ = 2^40 байт);

– -peti – показать вывод в пебибайтах (1 ПиБ = 2^50 байт);

– -si – вместо 1024 используют --mebi --si 1000. Например, --mebi --si равно --mega.

Например, чтобы отобразить результат в мегабайтах, следует выполнить:

```
free --mega
```

Чтобы постоянно отображать информацию о памяти на экране, нужно вызвать `free` с параметром -s(--seconds), за которым следует число, определяющее задержку.

Например, чтобы выдавать информацию о памяти каждые пять секунд, необходимо выполнить (рисунок 83):

```
free -s 5
```

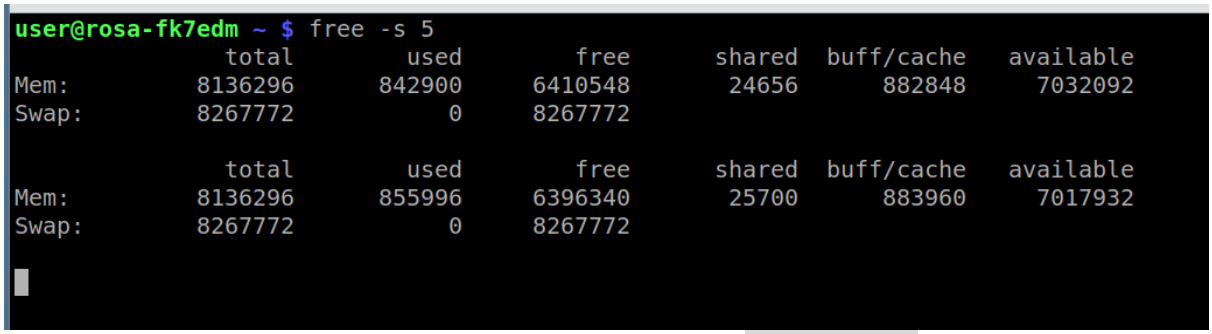


Рисунок 83 - Вывод команды `free -s 5`

Команда `free` будет продолжать отображать результат, пока не будет нажата комбинация клавиш `Ctrl+C`, что похоже на поведение команды `watch`.

Чтобы отображать результат определенное количество раз, используют параметр -c(--count) и целое число. Например:

```
free -s 5 -c 3
```

3.13.4.2 Команда htop

Команда `htop` – это интерактивный монитор Системы в терминале, который предоставляет детальную информацию о загрузке процессора,

использовании памяти, активности сети и других ресурсах. Вот несколько ключевых функций и использований `htop`:

– отображение активных процессов и их иерархии, а также загрузки CPU, памяти, дисков и сети;

– интерактивное управление процессами – изменение приоритета, приостановка, возобновление и завершение;

– визуальный мониторинг ресурсов с помощью цветных графиков и индикаторов;

– сортировка, поиск и фильтрация процессов по различным параметрам.

Команду `htop` можно запускать с правами обычного пользователя (рисунок 84):

```
htop
```

Также команду `htop` можно запускать и с привилегиями суперпользователя:

```
sudo htop
```

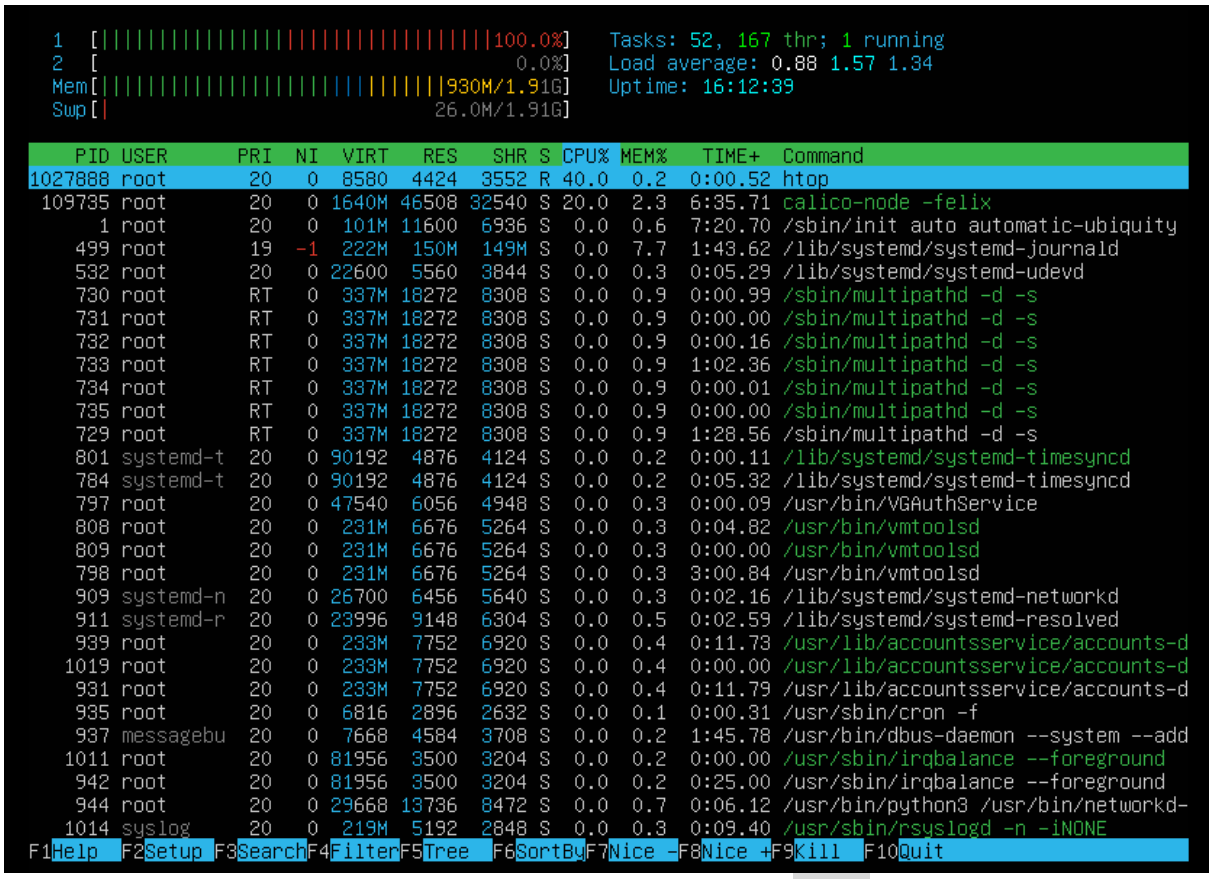


Рисунок 84 - Вывод команды `htop`

Основные параметры запуска `htop`:

– htop -u <имя_пользователя> – отображение только процессов указанного пользователя;

– htop -p <PID1,PID2,...> – показ только выбранных процессов по их PID;

– htop -s <поле> – сортировка списка по заданному полю (например, CPU%, MEM%, PID);

– htop -d <время> – установка интервала обновления экрана (в десятых долях секунды);

– htop -t – отображение процессов в виде дерева (Tree view);

– htop -C – отключение цветового оформления;

– htop -v – вывод информации о версии утилиты.

Горячие клавиши в интерактивном режиме:

– `F2` – настройки интерфейса;

– `F3` – поиск процесса;

– `F4` – фильтрация списка процессов;

– `F5` – переключение в древовидный режим;

– `F6` – выбор поля сортировки;

– `F9` – завершение выбранного процесса;

– `F10` – выход из программы.

3.13.4.3 Команда df

Команда `df` (disk free) используется для отображения информации о файловых системах, подключённых к Системе, включая их общий объём, использованное и свободное пространство, а также точки монтирования.

Базовый вызов команды (рисунок 85):

```
sudo df
```

```
user@rosa-fk7edm ~ $ sudo df
Файловая система 1K-блоков Использовано Доступно Использовано% Смонтировано в
devtmpfs              4096            0     4096            0% /dev
tmpfs              4068148            0  4068148            0% /dev/shm
tmpfs              1627260        12484  1614776            1% /run
/dev/sda2         30787492      8296948 20901296           29% /
tmpfs              4068148            0  4068148            0% /tmp
/dev/sda3         63793828        49776 60471092            1% /home
tmpfs               813628          112   813516            1% /run/user/500
```

Рисунок 85 - Вывод команды `df`

Наиболее используемые опции команды:

– -h – вывод в удобном для чтения формате (K, M, G);

– -H – аналогично -h, но с использованием степеней 1000, а не 1024;

– -T – отображение типа файловой системы;

– -i – отображение информации об inode (вместо байтов);

– -t <тип> – показать только файловые системы указанного типа (например, ext4, tmpfs);

– -x <тип> – исключить файловые системы заданного типа;

– --total – добавление строки с общим суммарным объёмом всех разделов;

– -a – вывод всех файловых систем, включая псевдо-файловые и те, где размер равен нулю;

– -P – использование стандартного формата POSIX (удобно при парсинге вывода скриптами);

– -hT – оценка загрузки дисков и поиск переполненных разделов;

– -h /имя каталога – вывод объема доступного пространства в каталоге.

3.13.4.4 Команда du

Команда `du` сообщает приблизительный объем дискового пространства, используемого данными файлами или каталогами. Это практически полезно для поиска файлов и каталогов, занимающих большие объемы дискового пространства.

Общий вид команды:

```
du [OPTIONS]... FILE...
```

Если данный FILE является каталогом, команда `du` будет суммировать использование диска для каждого файла и подкаталога в этом каталоге. Если FILE не указан, `du` сообщит об использовании диска в текущем рабочем каталоге.

При выполнении без какой-либо опции `du` отображает использование диска для данного файла или каталога и каждого из его подкаталогов в байтах.

Также можно передать команде `du` несколько файлов и каталогов в качестве аргументов:

```
du ~/Documents ~/Pictures
```

Если запустить `du` для файла или каталога, для которого нет прав, в выводе получится примерно следующее "du: невозможно прочитать каталог". В этой ситуации нужно будет добавить к команде `sudo`.

Часто используемые параметры для команды `du`:

– -a – сообщает об использовании дискового пространства каждым файлом в каталоге:

```
du -a /tmp/
```

– -h – отображает информацию о пространстве, занимаемом только данным каталогом, в удобочитаемом формате, например, чтобы получить общий размер /var/lib и всех его подкаталогов, нужно выполнить следующую команду:

```
sudo du -h /var/lib
```

Следует использовать команду sudo, потому что большинство файлов и каталогов внутри каталога /var/lib принадлежат пользователю root и не доступны для чтения обычным пользователям.

– -s – отображает только общий размер указанного каталога, а не подкаталогов (рисунок 86):

```
sudo du -sh /var
```

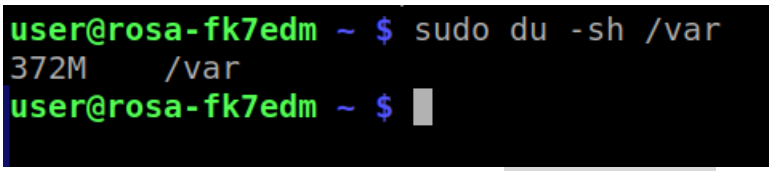


Рисунок 86 - Вывод команды du -sh /var

– -c – указывает команде du сообщать общую сумму, если требуется получить объединенный размер двух или более каталогов (рисунок 87):

```
sudo du -csh /var/log /var/lib
```

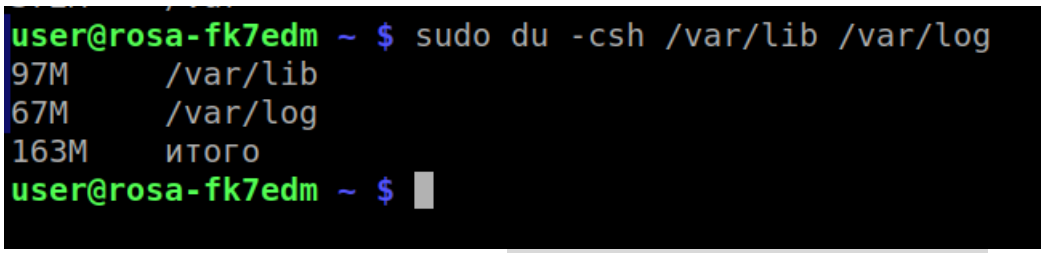


Рисунок 87 - Вывод команды du -csh /var/lib /var/log

– --max-depth – отображает использование диска подкаталогами n-уровня с указанием уровня подкаталогов, например, чтобы получить отчет о каталогах первого уровня, используют (рисунок 88):

```
sudo du -h --max-depth=1 /var/log
```

```
user@rosa-fk7edm ~ $ sudo du -h --max-depth=1 /var/log
4,0K    /var/log/sa
4,0K    /var/log/chrony
4,0K    /var/log/sudo-io
65M     /var/log/journal
4,0K    /var/log/ppp
8,0K    /var/log/cups
4,0K    /var/log/gdm
8,0K    /var/log/pluto
1,9M    /var/log/anaconda
24K     /var/log/samba
4,0K    /var/log/private
67M     /var/log
user@rosa-fk7edm ~ $
```

Рисунок 88 - Вывод команды `du -h --max-depth=1 /var/log`

– --apparent-size – по умолчанию утилита du повторно использует дисковое пространство, используемое каталогом или файлом; чтобы узнать очевидный размер файла (фактический объем данных в файле), используют переключатель (рисунок 89):

```
sudo du -sh --apparent-size /var/lib
```

```
user@rosa-fk7edm ~ $ sudo du -csh /etc/rc*
0       /etc/rc0.d
0       /etc/rc1.d
0       /etc/rc2.d
0       /etc/rc3.d
0       /etc/rc4.d
0       /etc/rc5.d
0       /etc/rc6.d
84K     /etc/rc.d
8,0K    /etc/rc_maps.cfg
0       /etc/rc.modules
92K     итого
user@rosa-fk7edm ~ $ sudo du -sh --apparent-size /var/lib
96M     /var/lib
user@rosa-fk7edm ~ $
```

Рисунок 89 - Вывод команды `sudo du -sh --apparent-size /var/lib`

`du` также позволяет использовать шаблон оболочки. Например, чтобы получить размер всех каталогов из /etc, начинающихся с "rc", нужно выполнить:

```
sudo du -csh /etc/rc*
```

3.13.4.5 Комбинации команд

Команду `du` можно комбинировать с другими командами с конвейерами.

Например, чтобы напечатать 5 самых больших каталогов внутри каталога `/var` нужно передать вывод `du` в команду `sort`, чтобы отсортировать каталоги по их размеру, а затем передать результат в команду `head`, которая распечатает только 5 верхних каталогов:

```
sudo du -h /var/ | sort -rh | head -5
```

В отличие от команды `df`, которая выводит информацию об использовании диска смонтированными ФС, команда `du` дает оценку дискового пространства, используемого данными файлами или каталогами.

3.14 Отладка и диагностика

Довольно часто приходится сталкиваться с тем, что программное обеспечение не работает должным образом. В разделе приведены наиболее простые инструменты и способы поиска ошибок в программах.

3.14.1 Консольный отладчик GDB

GDB – полноценный консольный отладчик для ОС который позволяет останавливать выполнение в произвольных точках, отслеживать состояние переменных, стека, памяти и регистров и т. д. Возможна удалённая отладка.

Запуск осуществляется с помощью команды:

```
gdb <имя исполняемого файла>
```

Также запуск можно осуществить с помощью команды:

```
gdb
```

После запуска команды можно указать файл для отладки:

```
file <имя исполняемого файла>
```

Еще можно подключиться к удалённому сеансу отладки:

```
target remote <адрес хоста>:<порт>
```

Также можно отлаживать уже запущенную программу (нужны права суперпользователя):

```
attach <pid>
```

Для запуска отлаживаемой программы используется команда:

```
run <аргументы>
```

где

– <аргументы> — параметры, которые передаются программе при запуске (аналогично тому, как они указываются при обычном запуске из терминала).

Если программа останавливается на установленной точке останова или при возникновении сигнала, GDB передаёт управление пользователю.

Для повторного запуска программы без выхода из GDB используется та же команда `run`, при этом предыдущие параметры можно не указывать — они сохраняются автоматически.

Точки останова можно установить командой:

```
break main.c:15
```

Или, если отлаживается, например, программа из одного файла, можно использовать следующую команду:

```
break 15
```

После остановки программы (например, на точке останова) для поэтапного анализа работы используются следующие команды:

– `step` – выполняет текущую строку и переходит внутрь вызываемой функции, если она присутствует;
– `next` – выполняет текущую строку без входа в вызываемую функцию, переходя к следующей строке после её выполнения;
– `continue` – возобновляет выполнение программы до следующей точки останова, получения сигнала или завершения работы программы;
– `quit` – завершает сеанс работы GDB и возвращает пользователя в терминал.

Для анализа значений переменных и выражений используется команда:

```
print <выражение>
```

Она позволяет вычислить и вывести значение любой переменной или выражения языка C, доступного в текущей точке выполнения программы.

Для просмотра исходного кода в контексте текущей строки применяется команда:

```
list
```

Она выводит фрагмент программы вокруг текущего места выполнения, что удобно при пошаговой отладке.

GDB наиболее эффективен при наличии исходных кодов и отладочной информации (файлов с символами).

Дополнительные сведения о возможностях отладчика приведены в справочной системе GDB:

```
info gdb
```

В некоторых случаях для анализа работы программы может быть удобнее использовать системные инструменты трассировки, такие как `strace`, которые позволяют отслеживать системные вызовы и сигналы без необходимости пошаговой отладки.

3.14.2 Диагностика и отладка процессов с помощью утилиты strace

`strace` – утилита, позволяющая отслеживать системные вызовы, используемые программой, и получаемые ей сигналы.

Запуск осуществляется с помощью одной из команд:

```
strace program
```

или

```
strace -p pid
```

Для отладки уже запущенной программы (нужны права суперпользователя).

Можно отлаживать многопроцессные программы, передав ключ `-f`. В этом случае будут печататься системные вызовы и сигналы не только родителя, но и детей, порождённых после присоединения к процессу. Можно указать файл, в который `strace` будет писать вывод:

```
strace -o log.txt program
```

Для системных вызовов `strace` печатает имя вызова, переданные ему аргументы и значение, которое он вернул. Можно ограничить количество отслеживаемых системных вызовов:

```
strace -e trace=open,close,read,write,fork,exec program
```

Так же можно выполнить команду для сигналов:

```
strace -e signal=!SIGTERM program
```

Можно отслеживать данные, читаемые или записываемые в файловые дескрипторы:

```
strace -e read=0,3 program
strace -e write=1,2,4 program
```

Кроме того, можно вмешиваться в работу процесса, например, подменяя результаты системных вызовов. Об этом подробнее можно прочитать в справочной документации strace по команде:

```
man 1 strace
```

3.14.3 Мониторинг и анализ состояния системы через псевдо-файловую систему /proc (procfs)

В ОС существует псевдофайловая система procfs, которая в большинстве дистрибутивов монтируется в общую ФС в каталог /proc. У данной ФС нет физического места размещения, нет блочного устройства, такого как жесткий диск. Вся информация, хранимая в данном каталоге, находится в оперативной памяти компьютера, контролируется ядром ОС и не предназначена для хранения файлов пользователя.Различную отладочную информацию можно также почерпнуть из procfs. Сведения о процессе с id pid можно найти в каталоге /proc/pid. Из такой информации можно выделить:

– Файл /proc/pid/exe является символической ссылкой на бинарный файл работающего процесса;

– Файл /proc/pid/status, содержащий различные данные о состоянии процесса, такие как:

– имя исполняемого файла;

– umask процесса;

– собственно состояние процесса;

– uid и gid процесса;

– количество потоков процесса.

– маски заблокированных, принимаемых и игнорируемых сигналов; маски представлены в виде 8 байт, среди которых бит с номером "i" отвечает сигналу с номером "i"; биты отсчитываются с единицы, справа налево;

– Файл /proc/pid/syscall содержит номера работающих в данный момент системных вызовов, вызванных из программы, и их аргументы;

– Файл /proc/pid/maps содержит список виртуальных страниц в адресном пространстве программы, файл /proc/pid/pagemap, их отображение в физические страницы;

– Файл /proc/pid/stack содержит трейс стека ядра;

– Каталог /proc/pid/fd содержит ссылки на файлы, соответствующие открытым в процессе файловым дескриптором;

– Каталог /proc/pid/fdinfo содержит файлы с информацией об этих дескрипторах;

– Файл /proc/pid/limits содержит таблицу лимитов для процесса.

Более полное представление о структуре procfs можно получить в документации:

```
man 5 proc
```

4 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

Основу классической модели аутентификации в ОС составляют файлы /etc/passwd и /etc/shadow, содержащие информацию об учетных записях пользователей и их паролях.

Модули PAM обращаются к этим файлам при проверке подлинности, поэтому понимание их структуры и назначения необходимо для администрирования системы безопасности.

4.1 Управление учётными данными пользователей

4.1.1 Настройка основных параметров пользователей в /etc/passwd/

/etc/passwd – это база данных на основе обычного текста, которая содержит информацию для всех учетных записей пользователей в Системе. Он принадлежит пользователю root и имеет права 644 (rw-r--r--). Файл может быть изменен только пользователем root или пользователями с привилегиями sudo и доступен для чтения всем пользователям Системы.

Следует избегать изменения файла /etc/passwd вручную без уверенности в своих действиях. Рекомендуется всегда использовать команду, предназначенную для этой цели.

Например, чтобы изменить учетную запись пользователя, используют команду `usermod`, а чтобы добавить новую учетную запись пользователя, используют команду `useradd`.

Файл /etc/passwd – это текстовый файл с одной записью в строке, представляющий учетную запись пользователя. Чтобы просмотреть содержимое файла, используют текстовый редактор или команду, например, `cat`:

```
cat /etc/passwd
```

Обычно первая строка описывает пользователя root, за ней следуют учетные записи системного и обычного пользователей. Новые записи добавляются в конец файла.

Каждая строка файла /etc/passwd содержит семь полей, разделенных запятыми:

```
rosa:x:1001:1001:rosa,,,:/home/user:/bin/bash
[--] [--] [--] [-----] [--------] [--------]
| | | | | | |
```

```
| | | | | | +-> 7. Login shell
| | | | | +----------> 6. Home directory
| | | | +--------------------> 5. GECOS
| | | +--------------------------> 4. GID
| | +------------------------------> 3. UID
| +-----------------------------------> 2. Password
+--------------------------------------> 1. Username
```

– **Имя пользователя** – Логин при входе в Систему. Каждое имя пользователя должно быть уникальной строкой на ПК. Максимальная длина имени пользователя ограничена 32 символами;

– **Пароль** – В старых Linux-подобных системах зашифрованный пароль пользователя хранился в /etc/passwd. В ОС для этого поля установлено значение "x" а пароль пользователя хранится в /etc/shadow;

– **UID** – Идентификатор пользователя – это номер, присвоенный каждому пользователю. Он используется ОС для обозначения пользователя

– **GID** – Номер идентификатора группы пользователя, относящийся к основной группе пользователя. Когда пользователь создает файл, группа файла устанавливается в эту группу. Обычно название группы совпадает с именем пользователя. Вторичные группы пользователей перечислены в /etc/groups**;**

– **GECOS** – Это поле содержит список значений, разделенных запятыми, со следующей информацией:

– полное имя пользователя или название приложения;
– номер комнаты;
– рабочий номер телефона;
– номер домашнего телефона;
– другая контактная информация.

– **Домашний каталог** – Абсолютный путь к домашнему каталогу пользователя. Он содержит файлы и конфигурации пользователя. По умолчанию домашние каталоги пользователей названы по имени пользователя и создаются в каталоге /home;

– **Оболочка входа** – Абсолютный путь к оболочке входа пользователя. Это оболочка, которая запускается, когда пользователь входит в Систему, по умолчанию – Bash.

4.1.2 Управление паролями в/etc/shadow/

/etc/shadow – это текстовый файл, содержащий информацию о паролях пользователей Системы. Он принадлежит пользователю root и теневой группе и имеет права 640 (rw-r-----).

Файл содержит по одной записи в каждой строке, каждая из которых представляет учетную запись пользователя. Можно просмотреть содержимое файла с помощью текстового редактора или такой команды, как `cat`:

```
sudo cat /etc/shadow
```

Обычно первая строка описывает пользователя root, за которым следуют учетные записи системного и обычного пользователя. Новые записи добавляются в конец файла.

Каждая строка файла /etc/shadow содержит девять полей, разделенных запятыми:

```
rosa:$6$.n.:17736:0:99999:7:::
[--] [----] [---] [---] ----
| | | | | |||+-----> 9. Unused
| | | | | ||+------> 8. Expiration date
| | | | | |+-------> 7. Inactivity period
| | | | | +--------> 6. Warning period
| | | | +----------> 5. Maximum password age
| | | +------------------> 4. Minimum password age
| | +--------------------> 3. Last password change
| +---------------------------> 2. Encrypted Password
+----------------------------------> 1. Username
```

– **Имя пользователя** – Строка, которую вводят при входе в систему; учетная запись пользователя, существующая в Системе.

– **Зашифрованный пароль** – Пароль использует формат $type$salt$hashed. $type – это алгоритм криптографического хеширования и может иметь следующие значения:

– 1 – 5 MDL;
– $2a$ – Blowfish;
– $2y$ – Eksblowfish;
– 5 – SHA-256;
– 6 – SHA-512.

Если в поле пароля есть звездочка "*" или восклицательный знак "!", пользователь не сможет войти в Систему, используя аутентификацию по паролю. Другие методы входа в Систему, такие как аутентификация на основе ключей или переключение на пользователя, по-прежнему разрешены.

– **Последнее изменение пароля** – это дата последней смены пароля. Число дней отсчитывается с 1 января 1970 года (дата эпохи).

– **Минимальный возраст пароля** – количество дней, которое должно пройти, прежде чем можно будет изменить пароль пользователя. Обычно он равен нулю, что означает отсутствие минимального возраста пароля;

– **Максимальный возраст пароля** – количество дней после смены пароля пользователя. По умолчанию этот номер установлен на 99999;

– **Период предупреждения** – количество дней до истечения срока действия пароля, в течение которых пользователя предупреждают о необходимости изменения пароля;

– **Период бездействия** – количество дней после истечения срока действия пароля пользователя до отключения учетной записи пользователя. Обычно это поле пусто;

– **Дата окончания срока** – дата, когда учетная запись была отключена;

– **Не используется** – это поле игнорируется. Оно зарезервировано для использования в будущем.

Файл/etc/shadow не следует редактировать вручную, если нет уверенности в своих действиях. Всегда используют команду, предназначенную для этой цели. Например, чтобы изменить пароль пользователя, используют команду `passwd`, а чтобы изменить информацию об устаревании пароля, используют команду `chage`.

Например, запись в файле /etc/shadow:

```
linuxize:$6$zHvrJMa5Y690smbQ$z5zdL...:18009:0:120:7:14:
```

содержит следующие сведения о пользователе:

– информация о пароле пользователя linuxize;

– пароль зашифрован с помощью SHA-512 (пароль усечен для лучшей читаемости);

– последний раз пароль изменялся 23 апреля 2019 г. 18009;

– нет минимального возраста пароля;

– пароль необходимо менять не реже, чем каждые 120 дней;

– пользователь получит предупреждающее сообщение за семь дней до истечения срока действия пароля;

– если пользователь не попытается войти в систему через 14 дней после истечения срока действия пароля, учетная запись будет отключена;

– срок действия учетной записи отсутствует.

4.2 Управление механизмами аутентификации и авторизации через модули PAM

PAM (Pluggable Authentication Modules) — это модульная архитектура, предназначенная для реализации механизмов аутентификации, авторизации,

управления сессиями и паролями в ОС РОСА "ХРОМ". Система PAM обеспечивает унифицированный программный интерфейс между прикладными программами (например, sshd, login, su, sudo, gdm и др.) и механизмами проверки подлинности пользователей.

Ключевое преимущество PAM заключается в возможности централизованного и гибкого управления политиками безопасности без необходимости изменения исходного кода прикладного программного обеспечения. Все изменения поведения системы аутентификации выполняются путём редактирования конфигурационных файлов PAM, что повышает управляемость и расширяемость Системы.

PAM реализуется как набор библиотек, подключаемых динамически и обрабатывающих запросы на вход пользователя. Сценарий выполнения зависит от результатов работы каждого модуля, конфигурации порядка их подключения и используемой логики (control flags).

4.2.1 Механизм работы PAM

Работа системы PAM строится вокруг понятия "стек аутентификации", в рамках которого выполняется последовательный вызов определённых модулей в установленном порядке. Для каждого прикладного сервиса (например, sshd, login, sudo, gdm-password) создаётся отдельный конфигурационный файл в каталоге /etc/pam.d/.

Каждая строка в конфигурационном файле имеет следующую структуру:

```
<тип> <контроль> <модуль> [аргументы]
```

где:

– **тип (type)** – указывает назначение модуля:

– auth – проверка подлинности пользователя (например, ввод пароля или PIN-кода);

– account – проверка политики доступа и допустимости входа (например, срок действия учётной записи, членство в группе);

– password – управление паролем (например, смена, проверка сложности, политика смены);

– session – инициализация и завершение пользовательской сессии (например, монтирование ресурсов, логирование активности).

– **контроль (control flag)** – определяет, как результат выполнения модуля влияет на общее решение (разрешить или запретить доступ):

– required – модуль должен выполниться успешно; при ошибке вход будет запрещён, даже если другие модули завершатся успешно;

– requisite – аналогично required, но при ошибке выполнение следующих модулей будет прекращено;

– sufficient – при успешном выполнении модуля дальнейшая проверка может быть пропущена, если это допускается конфигурацией;

– optional – результат выполнения модуля игнорируется, если он не является единственным в своей категории;

– include – подключение правил из другого PAM-файла;

– substack – подключение стека из другого файла с сохранением логики типа.

– **модуль** – путь к библиотеке или её имя (если библиотека находится в стандартном каталоге /lib/security/ или /lib64/security/);

– **аргументы** – параметры, передаваемые модулю (например, try_first_pass, nullok, debug, inactive=N и т.д.).

Пример записи:

```
auth required pam_unix.so try_first_pass nullok
```

Данная строка означает, что при аутентификации (auth) обязательно требуется успешное выполнение модуля pam_unix, использующего локальные пароли пользователей. Модуль будет использовать ранее введённый пароль (если он уже есть в стеке PAM) и разрешит вход без пароля, если поле пароля пустое (nullok).

В ОС РОСА "ХРОМ" используются два уровня конфигурации PAM:

– Общие шаблоны, которые располагаются в файлах:

– /etc/pam.d/system-auth;
– /etc/pam.d/password-auth.

Эти файлы содержат общие правила аутентификации, применяемые по умолчанию во всех сервисах через директивы include system-auth или include password-auth. Изменения, внесённые в эти файлы, распространяются на большинство сервисов.

– Специфические настройки для сервисов. Каждый сервис может переопределить или дополнить правила PAM в своём отдельном файле, например:

– /etc/pam.d/login – консольный вход (TTY);
– /etc/pam.d/sshd – вход по SSH;
– /etc/pam.d/sudo – выполнение команд от имени другого пользователя;
– /etc/pam.d/gdm-password – вход через графический интерфейс GDM.

При наличии противоречий между общими и локальными файлами приоритет имеет локальный файл конкретного сервиса.

Пример включения общих правил:

```
auth include system-auth
```

Эта директива указывает PAM использовать те же правила аутентификации, что и в файле /etc/pam.d/system-auth.

4.2.2 Конфигурация распространённых модулей PAM

4.2.2.1 Аутентификация через локальные пароли (модуль pam_unix.so)

Модуль pam_unix.so является базовым модулем для аутентификации и управления паролями с использованием локальной базы данных пользователей /etc/passwd и /etc/shadow.

Модуль pam_unix.so обеспечивает стандартную проверку пароля путём сравнения введённого значения с хешем, хранящимся в /etc/shadow. При смене пароля также используется модуль pam_unix.so, если в стеке PAM он подключён в секции "password". Модуль поддерживает использование различных параметров, позволяющих более гибко управлять политиками безопасности, например: ограничением на использование пустых паролей, принудительным использованием одного и того же пароля в цепочке модулей и др.

Основные параметры модуля:

– nullok – разрешение входа при пустом пароле;
– try_first_pass – попытка использовать уже введённый пароль из предыдущего модуля;
– use_authtok – обязательное использование пароля, полученного от предыдущего модуля (обычно используется в стеке password);
– shadow – чтение паролей из /etc/shadow (по умолчанию включено).

Пример использования:

```
auth        sufficient    pam_unix.so try_first_pass nullok
account     required      pam_unix.so
password    sufficient    pam_unix.so use_authtok
```

4.2.2.2 Управление переменными окружения при входе (модуль pam_env.so)

Модуль pam_env.so используется для задания и инициализации переменных окружения в сеансах пользователей при аутентификации через PAM. Он позволяет определить, какие переменные среды должны быть установлены, а также откуда они должны быть загружены.

Конфигурационные файлы модуля pam_env.so:

– /etc/security/pam_env.conf – основной файл, содержащий переменные окружения в формате "ключ=значение". Может содержать как статические переменные, так и значения, полученные из текущего окружения.

– /etc/environment – альтернативный источник переменных окружения. Он используется как вспомогательный механизм и поддерживает только простые объявления переменных без логики и условий.

Модуль pam_env.so выполняется до выполнения остальных PAM-модулей и позволяет задать окружение, которое будет использоваться в рамках сеанса пользователя. Это может быть полезно, например, для установки языковых локалей (LANG, LC_ALL), прокси-переменных, путей к пользовательским скриптам и т. д.

Конфигурационный файл /etc/security/pam_env.conf поддерживает:

– комментарии (начинаются с "#");
– подстановки переменных (например, ${home});
– управление значениями на основе текущего окружения.

4.2.2.3 Ограничение попыток входа (модуль pam_faillock.so)

Модуль pam_faillock.so отвечает за реализацию механизма блокировки учётной записи после нескольких неудачных попыток входа. Рекомендуется для защиты от атак перебора паролей.

Блокировка применяется как для локального входа (через TTY, GDM), так и для удалённого (через SSH), если PAM настроен соответствующим образом. Модуль сохраняет информацию о неудачных попытках входа и отслеживает их накопление для каждой учётной записи.

Блокировка через pam_faillock.so действует только на вход через PAM. Вызовы, обходящие PAM, не учитываются.

По умолчанию модуль использует файл /var/log/faillock для хранения состояния.

Модуль работает в двух режимах:

– preauth – анализирует количество предыдущих неудачных попыток до выполнения основной аутентификации;

– authfail – регистрирует неудачную попытку после того, как пользователь ввёл неверный пароль.

При достижении заданного количества попыток доступ блокируется на определённый промежуток времени (unlock_time) или до административной разблокировки.

Основные параметры:

– preauth – активация проверки до ввода пароля;

– authfail – регистрация неудачной попытки после неверного пароля;

– deny=N – максимальное количество попыток перед блокировкой (по умолчанию: 3);

– unlock_time=T – продолжительность блокировки в секундах;

– fail_interval=T – время, за которое считаются попытки (по умолчанию: 900);

– even_deny_root – распространение блокировки на пользователя root;

– audit – запись в системный журнал о блокировках и разблокировках;

– silent – скрытие вывода о количестве оставшихся попыток;

– local_users_only – применение только к локальным пользователям;

– no_log_info – отключение дополнительной информации в выводе.

Администратор может вручную удалить информацию о блокировках, тем самым разблокировав пользователя. Пример команды:

```
faillock --user <имя_пользователя> --reset
```

Просмотр статистики неудачных попыток:

```
faillock --user <имя_пользователя>
```

Пример конфигурации /etc/pam.d/system-auth для блокировки учетной записи пользователя после N неудачных попыток входа:

```
auth     required      pam_faillock.so preauth silent deny=5
unlock_time=900
auth     required      pam_unix.so
auth        [default=die]  pam_faillock.so  authfail  deny=5
unlock_time=900
account  required      pam_faillock.so
```

В данном примере при пяти неудачных попытках вход блокируется на 15 минут.

4.2.2.4 Ограничение ресурсов пользователей (модуль pam_limits.so)

Модуль pam_limits.so отвечает за применение ограничений на ресурсы пользователей (число процессов, размер памяти, количество открытых файлов и др.).

Ограничения применяются в соответствии с конфигурационными файлами limits.conf и дополнительными файлами из каталога /etc/security/limits.d/.

Ограничения вступают в силу при открытии новой сессии, поэтому необходимо убедиться, что pam_limits.so подключён в секции "session" соответствующего PAM-файла (например, login, sshd, su, gdm-password и др.).

Ограничения могут устанавливаться по следующим ресурсам:

– core – максимальный размер файла core dump (в блоках);
– data – максимальный размер сегмента данных процесса;
– fsize – максимальный размер создаваемого файла (в блоках);
– memlock – максимальный объём заблокированной памяти;
– nofile – максимальное количество открытых файлов;
– nproc – максимальное количество процессов пользователя;
– rss – максимальный объём оперативной памяти;
– stack – максимальный размер стека;
– cpu – лимит времени процессора (в минутах);
– maxlogins – максимальное количество одновременных логинов;
– priority – приоритет (nice) запускаемых процессов;
– locks – максимальное количество блокировок файлов;
– sigpending – максимальное количество ожидающих сигналов;
– msgqueue – максимальный объём POSIX-очередей сообщений;
– nice – максимально допустимое значение nice;
– rtprio – максимальный приоритет реального времени.

Каждая строка в файлах имеет следующий формат:

```
<домен> <тип> <ресурс> <значение>
```

– `<домен>` – имя пользователя, группа (@groupname) или спец. символ * (все пользователи);
– `<тип>` – soft (мягкое ограничение), hard (жёсткое ограничение) или "-" (оба сразу);
– `<ресурс>` – название ограничиваемого ресурса;
– `<значение>` – числовое значение ограничения или unlimited.

Настройки ограничений применяются со следующим приоритетом и порядком:

– настройки из /etc/security/limits.d/*.conf дополняют или переопределяют правила из limits.conf при совпадении домена и ресурса;

– приоритет имеют правила, указанные позже (например, в алфавитном последнем файле из limits.d);

– если для одного пользователя задано несколько значений одного ресурса, используется более строгое (жёсткое) ограничение.

Некоторые ограничения (например, nofile) могут быть переопределены средствами оболочки или системным демоном (например, systemd user services). В таких случаях требуется дополнительная настройка через systemd (директивы LimitNOFILE= и т.п.).

При отладке рекомендуется использовать команду `ulimit -a` для просмотра текущих применённых ограничений к сессии.

Примеры конфигурации модуля:

– Ограничение количества процессов:

```
@developers  soft  nproc  2048
@developers  hard  nproc  4096
```

– Ограничение количества одновременно открытых файлов:

```
user1  -  nofile  1024
```

– Принудительное окончание сессии и ограничение пользователя одной активной сессией (например, для терминального сервера). Изменения вносятся в файл /etc/security/limits.d/10-maxlogins.conf:

```
* hard maxlogins 1
```

4.2.2.5 Проверка условий доступа (модуль pam_succeed_if.so)

Модуль pam_succeed_if.so используется для условного выполнения последующих или текущих PAM-модулей в зависимости от заданных условий. Применяется для реализации гибкой логики доступа, например, разрешения или запрета входа определённым группам пользователей, пользователям с конкретными идентификаторами (UID), оболочками (shell), типами сервисов и т.д. Это особенно полезно при создании дифференцированных правил аутентификации в различных сценариях: GDM, TTY, SSH и других.

Модуль pam_succeed_if.so не выполняет саму аутентификацию, а лишь влияет на ход выполнения PAM-стека, пропуская или блокируя модули по заданным условиям.

Основные параметры модуля:

– user ingroup <имя_группы> — проверка, состоит ли пользователь в указанной группе;

– uid < N / uid >= N — проверка числового идентификатора пользователя (например, для различия между системными и обычными пользователями);

– shell = /bin/bash — проверка используемой оболочки пользователя;

– service = gdm — проверка PAM-сервиса (имя PAM-конфигурационного файла), в рамках которого вызывается модуль.

Модуль поддерживает как положительные, так и отрицательные условия, а также логические операторы. Для управления логикой перехода в стеке используются параметры управления, например:

– [default=1 success=ignore] — при выполнении условия переход к следующему модулю, в противном случае — пропуск следующего модуля;

– [success=ok default=bad] — возврат успешного статуса при выполнении условия, иначе — неудача.

Пример использования для ограничения входа только пользователям, состоящим в группе tokenlogin, используется следующая строка в PAM-конфигурации:

```
    auth [default=1 success=ignore] pam_succeed_if.so user ingroup\ tokenlogin
    auth requisite pam_deny.so
```

В данном примере, если пользователь не входит в группу tokenlogin, переход к следующей строке будет невозможен, выполнение pam_deny.so завершится отказом в доступе.

Данное правило может быть добавлено, например, в начало файла /etc/pam.d/system-auth или /etc/pam.d/login.

4.2.2.6 Аутентификация с использованием токенов и смарт-карт (модуль pam_p11.so)

Модуль pam_p11.so предназначен для аутентификации пользователей с использованием аппаратных токенов (по стандарту PKCS#11). Используется в рамках реализации двухфакторной аутентификации в ОС РОСА "ХРОМ". pam_p11.so позволяет выполнять проверку подлинности за счёт криптографической подписи, формируемой закрытым ключом, хранящимся на токене.

Особенности работы модуля:

– Модуль взаимодействует с аппаратным токеном напрямую через PAM и не зависит от наличия службы sshd или ключей OpenSSH.

– Применяется как для локального входа, так и в составе универсальных PAM-стеков, например, для GDM, SSH, TTY и других служб.

– В сценарии двухфакторной аутентификации модуль инициирует запрос PIN-кода от токена и проверяет криптографическую подпись случайной последовательности, тем самым реализуя второй фактор.

Подробные инструкции по подготовке токена, настройке целевой и рабочей станции, а также интеграции pam_p11.so с различными сервисами приведены в разделе 4.3 Настройка двухфакторной аутентификации.

4.2.3 Блокирование неактивных учётных записей

Для обеспечения безопасности и минимизации риска использования забытых или устаревших учётных записей модуль pam_lastlog позволяет реализовать блокирование входа пользователей, не осуществлявших вход в Систему в течение определённого количества дней.

4.2.3.1 Блокирование при входе через GDM

Для включения блокировки неактивных учётных записей при входе через графический менеджер входа (по умолчанию используется GDM) требуется внести изменения в файл /etc/pam.d/password-auth. Для этого необходимо:

а) открыть файл для редактирования с правами суперпользователя:

```
sudo nano /etc/pam.d/password-auth
```

добавить в него строку:

```
auth required pam_lastlog.so inactive=90
```

между строками:

```
auth required pam_env.so
```

и:

```
auth sufficient pam_unix.so try_first_pass nullok
```

также добавить строку:

```
account required pam_lastlog.so inactive=90
```

перед строкой:

```
account required pam_unix.so
```

Параметр inactive=90 указывает, что вход будет заблокирован, если последний вход пользователя был выполнен более 90 суток назад. При необходимости значение может быть изменено.

4.2.3.2 Блокирование при входе через терминал (TTY, SSH)

Для включения аналогичного механизма при входе через терминал или SSH необходимо внести аналогичные изменения в файл /etc/pam.d/system-auth. Для этого необходимо:

а) открыть файл для редактирования:

```
sudo nano /etc/pam.d/system-auth
```

добавить в файл строку:

```
auth required pam_lastlog.so inactive=90
```

между строками:

```
auth required pam_env.so
```

и:

```
auth sufficient pam_unix.so try_first_pass likeauth nullok
```

также добавить строку:

```
account required pam_lastlog.so inactive=90
```

перед строкой:

```
account required pam_unix.so
```

Данная конфигурация блокирует доступ к Системе пользователям, чья учётная запись не использовалась в течение указанного срока.

4.2.3.3 Просмотр справки по модулю pam_lastlog

Для получения дополнительной информации о возможных параметрах модуля pam_lastlog рекомендуется использовать встроенную справку:

```
man pam_lastlog
```

4.2.4 Отладка и проверка работы PAM

Корректная работа PAM имеет критическое значение для обеспечения доступа к Системе. Ошибки в конфигурации PAM-файлов могут привести к блокировке входа для всех пользователей, включая администратора. Поэтому

любые изменения в конфигурации должны сопровождаться обязательной проверкой и, по возможности, тестироваться в отдельной сессии.

4.2.4.1 Просмотр системного журнала

Большинство PAM-модулей, включая pam_unix, pam_faillock, pam_lastlog, pam_p11, записывают диагностические сообщения в системный журнал. Для просмотра сообщений, связанных с PAM, рекомендуется использовать команду:

```
sudo journalctl -b | grep pam
```

Для анализа конкретного модуля, например, pam_p11, используется команда:

```
sudo journalctl -b | grep pam_p11
```

Параметр -b ограничивает вывод сообщениями текущей загрузки Системы.

4.2.4.2 Проверка конфигурации и тестирование изменений

Перед внесением изменений в основные PAM-файлы (например, /etc/pam.d/system-auth, /etc/pam.d/password-auth, /etc/pam.d/sshd) рекомендуется, предварительно обязательно создав резервную копию изменяемого файла:

```
sudo cp /etc/pam.d/system-auth /etc/pam.d/system-auth.bak
```

а) создать отдельную тестовую сессию (например, новую вкладку в терминале TTY или подключение по SSH от другого пользователя), не завершая активную административную сессию;

б) выполнить проверку работы входа или сервиса с новой конфигурацией.

Если вход становится невозможен, следует восстановить резервную копию из открытой сессии:

```
sudo cp /etc/pam.d/system-auth.bak /etc/pam.d/system-auth
```

4.2.4.3 Использование временного доступа через root-tty

Для минимизации рисков полной потери доступа к Системе рекомендуется:

– оставлять root-доступ через терминал TTY активным (файл /etc/pam.d/login);

– не изменять правила PAM для sudo, su и login одновременно;

– иметь доступ к Системе через физическую консоль или загрузку в однопользовательском режиме (recovery mode), если доступ по SSH будет заблокирован.

4.2.4.4 Примеры отладочной информации от модулей

Для повышения информативности можно включить режим отладки в отдельных модулях:

– В модуле pam_unix:

```
auth required pam_unix.so debug
```

– В модуле pam_faillock:

```
auth required pam_faillock.so preauth silent deny=5\
unlock_time=900 debug
```

Сообщения с пометкой pam_unix(debug) или pam_faillock(debug) будут отображаться в journalctl.

4.2.4.5 Проверка порядка выполнения модулей

Для анализа последовательности выполнения PAM-модулей можно использовать повышенное логирование, активируемое на уровне службы (например, sshd). Для этого в файле /etc/ssh/sshd_config необходимо добавить или изменить параметр:

```
LogLevel DEBUG
```

После этого нужно перезапустить службу:

```
sudo systemctl restart sshd
```

В journalctl будут отображаться дополнительные сведения о ходе аутентификации.

4.3 Настройка двухфакторной аутентификации

Настройка двухфакторной (многофакторной) аутентификации в ОС выполняется с использованием программного модуля pam_p11 и аппаратного криптографического устройства (токена) Рутокен ЭЦП PKI.

Рассматривается разделение на рабочую станцию администратора – устройство, с которого выполняется настройка токена, и целевую станцию – устройство, на котором осуществляется вход в ОС с использованием второго фактора. В некоторых случаях обе роли могут быть совмещены на одном устройстве.

Первым фактором аутентификации выступает наличие устройства с уникальным закрытым ключом, вторым фактором – знание PIN-кода, ассоциированного с данным устройством.

4.3.1 Подготовка рабочих станций

Подготовка рабочих станций осуществляется в несколько этапов:

а) Для Систем на базе платформы ROSA2021.1 и новее требуется установка следующих пакетов:

```
sudo dnf install task-smartcards pam_p11
```

При установке пакета task-smartcards автоматически создаётся группа tokenlogin. Пользователи, входящие в данную группу, проходят аутентификацию с использованием второго фактора. Остальные пользователи проходят аутентификацию по паролю;

б) После установки пакета task-smartcards необходимо выполнить активацию службы pcscd.socket с помощью команды:

```
sudo systemctl enable --now pcscd.socket
```

В версиях ОС, начиная с 12.3, данная служба активирована по умолчанию. Повторная активация не приведёт к негативным последствиям.

в) Далее необходимо провести проверку и при необходимости настройку политики polkit. Пакет pcsc-lite содержит политику org.debian.pcsc-lite.policy, разрешающую доступ к токенам всем пользователям. При необходимости допускается создание пользовательской политики в каталоге /etc/polkit-1/. В большинстве случаев дополнительная настройка не требуется.

4.3.2 Подготовка токена Рутокен

Подготовка токена осуществляется в несколько этапов:

а) Подключение устройства. Токен Рутокен ЭЦП PKI должен быть подключён к рабочей станции администратора. Устройство должно определяться в Системе, например:

```
Bus 001 Device 005: ID 0a89:0030 Aktiv Rutoken ECP
```

б) Очистка данных с токена производится с помощью команды:

```
pkcs15-init --erase-card -p rutoken_ecp
```

Команда приводит к полной очистке устройства и выполняется только уполномоченным администратором.

в) Создание нового контейнера PKCS#15 осуществляется командой:

```
    pkcs15-init --create-pkcs15 --so-pin "<PIN администратора>"
" --so-puk ""
```

г) Присвоение контейнеру пользовательского PIN (например, 12345678) и PIN администратора (например, 87654321):

```
    pkcs15-init --store-pin --label "User PIN" --auth-id 02 –pin\
"12345678" --puk "" --so-pin "<PIN 87654321>" --finalize
```

д) Генерация ключевой пары (RSA длиной 2048 бит) с помощью команды:

```
    pkcs15-init -G rsa/2048 --auth-id 02 --id 42 --label "SSH\
User's Key" --public-key-label "SSH User's Public Key"
```

е) При выполнении команды будет запрошен PIN пользователя:

```
    User PIN [User PIN] required. Please enter User PIN [User\
PIN]:
```

Необходимо ввести заданный ранее пароль – 12345678.

4.3.3 Извлечение и перенос публичного ключа

При необходимости можно использовать токен в качестве второго фактора аутентификации при входе по SSH. Описанные далее действия можно выполнить на любой из станций.

Для использования токена при входе по SSH необходимо извлечь публичный ключ:

```
    ssh-keygen -D /usr/lib64/opensc-pkcs11.so -I 0:42 >> key.pub
```

Содержимое файла key.pub требуется добавить в файл ~/.ssh/authorized_keys соответствующего пользователя на целевой станции.

Файл ~/.ssh/authorized_keys используется PAM-модулем pam_p11 независимо от наличия и настроек сервера OpenSSH.

4.3.4 Подготовка целевой станции

Подготовка целевой станции осуществляется в несколько этапов.

На целевой станции требуется установка модуля pam_p11:

```
    sudo dnf install pam_p11
```

Добавление пользователей в группу tokenlogin (группа автоматически создается при установке пакета task-smartcards) выполняется следующими действиями:

а) Добавление локальных пользователей:

```
sudo usermod -a -G tokenlogin user
```

где вместо user используется имя целевого пользователя.

б) Добавление доменных пользователей с использованием механизма сопоставления групп, который реализуется с помощью пакета libnss-role.

Данный механизм позволяет считать члена одной группы (например, из домена) одновременно членом другой — локальной.

Предполагается, что на контроллере домена (Microsoft Active Directory / Samba AD либо FreeIPA) имеется группа с именем Token Login. В случае корректного включения станции с установленной ОС РОСА "ХРОМ" в домен, Система будет видеть принадлежность доменного пользователя к данной группе.

Для активации механизма сопоставления необходимо выполнить установку пакета:

```
sudo dnf install libnss-role
```

В процессе установки модуль role будет автоматически добавлен в строку "group" конфигурационного файла /etc/nsswitch.conf.

Далее следует определить соответствие между доменной группой и локальной:

```
sudo roleadd 'Token Login' tokenlogin
```

В результате в файл /etc/role будет добавлена соответствующая запись. После этого при выполнении команды `id` доменного пользователя в выводе отразится его принадлежность как к группе Token Login, так и к локальной группе tokenlogin.

4.3.5 Список ключей пользователя

Для обеспечения возможности входа в Систему с использованием второго фактора (токена) необходимо добавить открытый ключ, извлечённый из токена, в файл ~/.ssh/authorized_keys соответствующего пользователя.

При подключённом токене требуется выполнить следующую команду:

```
ssh-keygen -D /usr/lib64/opensc-pkcs11.so -I 0:42
```

Результат выполнения команды представляет собой строку, содержащую открытый ключ. Данный ключ необходимо внести в файл ~/.ssh/authorized_keys пользователя, которому предполагается разрешение входа с использованием токена.

Процедура выполняется от имени данного пользователя и включает следующие шаги:

а) создание каталога .ssh (при отсутствии):

```
mkdir --mode=700 -p ~/.ssh
```

б) открытие файла ~/.ssh/authorized_keys на редактирование:

```
nano ~/.ssh/authorized_keys
```

вставка строки с открытым ключом, полученным ранее, и сохранение файла.

Файл ~/.ssh/authorized_keys используется PAM-модулем pam_p11 как при удалённом входе по SSH, так и при локальном входе через терминал. Наличие соответствующего открытого ключа в этом файле является обязательным условием для успешной аутентификации с использованием токена.

4.3.6 Создание общего файла настроек PAM

Для организации централизованной конфигурации аутентификации с использованием токена рекомендуется создать отдельный файл /etc/pam.d/p11, содержащий настройки, специфичные для работы PAM-модуля pam_p11.

Создание и редактирование файла выполняется следующим образом:

```
sudo nano /etc/pam.d/p11
```

В файл необходимо внести следующую конфигурацию:

```
auth       required    pam_env.so
#auth     required    pam_faillock.so preauth silent deny=5\
unlock_time=900
auth      required    pam_p11.so /usr/lib64/opensc-pkcs11.so
#auth        required     pam_faillock.so authfail deny=5\
unlock_time=900
password   optional    pam_p11.so /usr/lib64/opensc-pkcs11.so
```

После внесения изменений файл следует сохранить (в редакторе nano для этого используется комбинация клавиш `Ctrl+O`), затем нажать клавишу `Enter` и выйти по комбинации клавиш `Ctrl+X`.

Если планируется использование модуля pam_faillock для защиты от подбора PIN-кода, необходимо удалить символ решетки (#) в начале соответствующих строк, тем самым активировав их (раскомментировать).

В дальнейшем при необходимости следует указать подключение данного файла (стека p11) в других PAM-конфигурациях, например, в файлах login, sshd или gdm-password, посредством директивы `substack p11`.

4.3.6.1 Особенности работы pam_p11

Если ни один из ранее вызванных модулей PAM не запросил пароль, pam_p11 инициирует его запрос и использует введённый PIN-код для доступа к закрытому ключу на токене. Если же пароль уже получен предыдущим модулем, pam_p11 использует его как PIN-код. В конфигурациях, приведённых в данном разделе руководства, модуль pam_p11 всегда вызывается первым и самостоятельно инициирует ввод PIN-кода.

4.3.7 Расширенная настройка

Далее в разделах 4.3.7.1-4.3.7.6 рассматриваются типовые примеры настройки PAM для использования второго фактора аутентификации. При необходимости допускается реализация более сложных и специфичных конфигураций PAM-модулей.

Для ознакомления с официальной справочной информацией по формату конфигурационных файлов PAM рекомендуется использовать встроенную документацию. Для этого нужно использовать команду:

```
man pam.conf
```

4.3.7.1 Просмотр логов

После выполнения попытки входа в Систему возможно получение диагностической информации о работе PAM-модулей, в том числе pam_p11, посредством системного журнала:

```
sudo journalctl -b | grep pam_p11
```

Вывод ограничивается сообщениями текущего сеанса загрузки Системы. При необходимости фильтрация может быть адаптирована под другие модули или условия.

4.3.7.2 Настройка входа в TTY

Настройка, при которой токен используется исключительно при входе в Систему через терминал TTY (переключение с помощью комбинации клавиш `Alt+Ctrl+F[N]`), выполняется в конфигурационном файле /etc/pam.d/login.

Для внесения изменений необходимо открыть файл с правами суперпользователя:

```
sudo nano /etc/pam.d/login
```

Затем нужно внести следующие изменения в файл:

а) Замена строки auth include system-auth на блок:

```
    auth  [default=1  success=ignore]  pam_succeed_if.so  user\
ingroup\ tokenlogin
    auth substack p11
    auth  [default=ignore  success=1]  pam_succeed_if.so  user\
ingroup\ tokenlogin
    auth substack system-auth
```

б) Замена строки password include system-auth на блок:

```
    password [default=1 success=ignore] pam_succeed_if.so user\
ingroup tokenlogin
    password substack p11
    password [default=ignore success=1] pam_succeed_if.so user\
ingroup tokenlogin
    password substack system-auth
```

в) После сохранения изменений рекомендуется выполнить вход через терминал TTY под пользователем, для которого ранее был настроен ключ SSH. При возникновении проблем следует обратиться к журналу Системы, как описано в п. 4.3.7.1.

4.3.7.3 Настройка входа через GDM

Для настройки входа через графический дисплей-менеджер GDM требуется изменение файла /etc/pam.d/gdm-password.

Необходимо открыть файл на редактирование с правами суперпользователя (например, с помощью редактора nano):

```
    sudo nano /etc/pam.d/gdm-password
```

Далее нужно внести в файл /etc/pam.d/gdm-password следующие изменения:

а) закомментировать строки (добавить символ решетки (#) в начало строки):

```
    auth        substack      password-auth
    auth        optional      pam_gnome_keyring.so
```

добавить после закомментированных строк следующий блок:

```
    auth  [default=1  success=ignore]  pam_succeed_if.so  user\
ingroup tokenlogin
    auth substack p11
```

```
    auth  [default=ignore  success=1]  pam_succeed_if.so  user\
ingroup tokenlogin
    auth substack password-auth
```

аналогично закомментировать следующие строки:

```
    password    substack       password-auth
    -password   optional       pam_gnome_keyring.so use_authtok
```

Следует обратить внимание, что отключение модуля pam_gnome_keyring необходимо в данном случае, поскольку его назначение – автоматическая разблокировка пользовательского хранилища ключей при совпадении пароля от хранилища с системным паролем пользователя – теряет смысл при использовании PIN-кода и токена.

добавить после них блок:

```
    password [default=1 success=ignore] pam_succeed_if.so user\
ingroup tokenlogin
    password substack p11
    password [default=ignore success=1] pam_succeed_if.so user\
ingroup tokenlogin
    password substack system-auth
```

сохранить изменения в файле и выполнить вход в Систему через GDM. При корректной настройке вместо стандартного пароля будет запрошен PIN-код от токена.

4.3.7.4 Настройка входа по SSH

При входе в Систему по SSH с использованием токена в качестве второго фактора аутентификации токен должен быть физически подключён к рабочей станции, с которой производится подключение, а на целевой станции должен быть зарегистрирован только открытый ключ.

Содержимое открытого ключа, извлечённого с токена, должно быть добавлено в файл ~/.ssh/authorized_keys соответствующего пользователя на целевой станции. Порядок извлечения ключа приведён в п. 4.3.3 Извлечение и перенос публичного ключа.

4.3.7.5 Вход по SSH с вводом PIN-кода и пароля

Настройка целевой станции

Рассматривается конфигурация, в которой при входе по SSH пользователь сначала вводит PIN-код токена, а затем – свой пароль. В данном варианте PAM-модуль pam_p11 не требуется.

Для активации данного режима необходимо запретить вход по паролю, разрешив только комбинацию открытого ключа и интерактивной аутентификации, командой:

```
    echo "AuthenticationMethods publickey,keyboard-interactive"\
| sudo tee -a /etc/ssh/sshd_config
```

Также необходимо убедиться, что параметр UsePAM в том же файле установлен в значение "yes" (данное значение задано по умолчанию в ОС):

```
    UsePAM yes
```

При необходимости можно скорректировать значение и перезапустить службу SSH:

```
    sudo systemctl restart sshd
```

Вход с рабочей станции

На рабочей станции, с которой выполняется подключение, должен быть установлен и подключён токен. Подключение осуществляется по следующей команде:

```
    ssh    -I    /usr/lib64/opensc-pkcs11.so    -p    <порт>\
<пользователь>@<ip\
    -адрес>
```

где параметры:

– <порт> – номер порта, на котором запущен sshd на целевой станции (по умолчанию – 22);

– <пользователь> – имя пользователя на целевой станции;

– <ip-адрес> – IP-адрес целевой станции.

После выполнения команды будут последовательно запрошены:

– PIN-код от токена;

– пароль пользователя.

4.3.7.6 Вход по SSH только с вводом PIN-кода

Настройка целевой станции

В данной конфигурации используется модуль pam_p11, а пользователь выполняет вход исключительно по PIN-коду без ввода системного пароля. Для этого пользователю нужно:

а) открыть на редактирование PAM-конфигурацию для sshd:

```
sudo nano /etc/pam.d/sshd
```

б) закомментировать строку (добавить символ решетки (#) в начало строки):

```
auth       include      system-auth
```

в) вставить вместо неё блок:

```
auth [default=1 success=ignore] pam_succeed_if.so user ingroup\ tokenlogin
auth substack p11
auth [default=ignore success=1] pam_succeed_if.so user ingroup\ tokenlogin
auth substack system-auth
```

аналогично закомментировать строку:

```
password   include  system-auth
```

вставить вместо неё блок:

```
password [default=1 success=ignore] pam_succeed_if.so user\ ingroup tokenlogin
password substack p11
password [default=ignore success=1] pam_succeed_if.so user\ ingroup tokenlogin
password substack system-auth
```

убедиться, что параметр UsePAM в /etc/ssh/sshd_config установлен в значение "yes".

Вход с рабочей станции

Токен должен быть подключён к рабочей станции, с которой производится вход. Подключение выполняется командой:

```
ssh -I /usr/lib64/opensc-pkcs11.so -p <порт>\ <пользователь>@<ip\
-адрес>
```

После этого будет запрошен только PIN-код. При корректной настройке вход будет выполнен без запроса системного пароля пользователя.

4.4 Управление пользователями в локальной сети с FreeIPA

Для централизованной аутентификации пользователей в локальной сети в ОС применяется технология IPA (FreeIPA). IPA – это интегрированное решение для управления идентификацией, аутентификацией и авторизацией ОС РОСА "ХРОМ" в сети. Этот сервер предоставляет централизованный контроль доступа и обеспечивает безопасность идентификации пользователей, компьютеров и служб в корпоративной среде.

4.4.1 Настройка сервера домена IPA

Для работы сервера IPA необходимо настроить сеть на сервере и имя хоста.

В качестве примера используются следующие данные для хоста IPA-сервера:

– имя хоста – dc1;
– имя домена – rosa.lan;
– IP-адрес IPA-сервера – 192.168.1.6;
– маска подсети, в которой будут использоваться IPA-сервер и рабочие станции – 255.255.255.0;
– шлюз по умолчанию – 192.168.1.1;
– DNS-сервер – 192.168.1.1;
– DNS-поиск домена – rosa.lan .

Алгоритм настройки сети:

а) установить имя сервера, например:

```
sudo hostnamectl dc1.rosa.lan
```

б) создать для сервера статический IP-адрес, используя интерфейс nmtui;
в) перезагрузить ОС, чтоб изменения вступили в силу.

После перезагрузки и входа в Систему необходимо установить пакет IPA-сервера:

```
sudo dnf install ipa-server
```

Далее нужно запустить его инсталляцию командой:

```
sudo ipa-server-install
```

В процессе установки сервера следует придерживаться следующих инструкций при запросе установщика:

а) о необходимости настройки DNS-сервера на IPA-сервере ответить "YES";

б) о наименовании хоста, домена и realm ввести данные, которые были рассмотрены для примера в начале раздела, а именно: dc1.rosa.lan, rosa.lan и ROSA.LAN;

в) о логинах и паролях для сервера и административного IPA-аккаунта ввести логин по умолчанию – admin, запрашиваемый пароль и запомнить их;

г) о DNS-сервере нужно ответить на следующие вопросы:

1) Do you want to configure DNS forwarders? – ответить "YES", чтобы сконфигурировать DNS-серверы для форвардинга;

2) Do you want to configure these servers as DNS forwarders? – ответить "YES", чтобы сконфигурировать найденный DNS-сервер как сервер для форвардинга;

3) Enter an IP address for a DNS forwarder, or press Enter to skip: – нажать клавишу `Enter`, чтобы не вписывать дополнительные DNS-сервера для форвардинга., или вписать IP-адреса DNS-серверов для форвардинга;

4) Do you want to search for missing reverse zone? – ответить "YES", чтобы найти обратную зону DNS;

5) Do you want to configure chrony with NTP server or pool address? – ответить "NO", чтобы не конфигурировать сервер chrony, так как устраивает его конфигурация по умолчанию.

д) при выводе конфигурационной информации (запроса о подтверждении правильности конфигурирования) для дальнейшей установки IPA-сервера ответить "YES", если все правильно сконфигурировано.

После этого шага начнется инсталляция IPA-сервера на ПК.

Если установка прошла успешно, установщик IPA-сервера выведет конечную информацию об установленном сервере и его сервисах.

Далее необходимо проверить работоспособность IPA-служб командой:

```
ipactl status
```

Если в выводах команды везде фигурирует RUNNING, значит все сервисы запущены и работают.

Следующим и последним шагом установки необходимо инициализировать аккаунт IPA-администратора – admin:

```
kinit admin
```

Далее нужно ввести пароль, который был введен ранее при установке IPA.

Установка может занять значительное время. Для инсталляции и работы IPA-сервера рекомендуется не менее 4 Гб оперативной памяти.

Далее управлять установленным IPA-сервером можно через web-интерфейс: в данном случае по адресу dc1.rosa.lan.

В открывшемся веб-интерфейсе FreeIPA-сервера нужно ввести логин (admin) и пароль, чтобы войти под учетной записью администратора.

После успешного входа откроется веб-интерфейс IPA-сервера, в котором будет производиться вся дальнейшая работа с сервером.

4.4.1.1 Создание пользователей

Чтобы добавить пользователей на сервере IPA, нужно перейти в веб-интерфейс сервера. Далее необходимо перейти в раздел "Идентификация → Активные пользователи", заполнить все необходимые параметры и нажать кнопку Добавить.

Следует обратить внимание, что в поле "Имя учетной записи" логин будущего пользователя должен быть введен латинскими буквами.

Информация, вводимая в поля "Имя" и "Фамилия" будущего пользователя, носит только справочную информацию и не влияет на дальнейшую работу.

В выпадающем списке поля "ID группы" необходимо выбрать группу editors.

Для добавления каждого нового пользователя необходимо пройти аналогичные шаги. Все вновь созданные пользователи появятся в разделе "Активные пользователи".

4.4.1.2 Настройка сети

Для настройки сети IPA-сервера нужно перейти в веб-интерфейс управления сервером. Далее в меню "Параметры Системы → Соединения" нужно выбрать соединение и вкладку "IPv4" (в правой части экрана) и задать следующие параметры:

– из выпадающего списка в параметре "Метод" выбрать "Вручную";
– в поле "DNS серверы" – IP-адрес IPA сервера 192.168.1.1;
– в поле "Домены поиска" – rosa.lan;
– добавить статические адреса:

  – 192.168.1.58 – это IP-адрес рабочей станции;
  – 255.255.255.0 – маска подсети;
  – 192.168.1.1 – шлюз.

По завершении нужно нажать кнопку Применить.

Далее необходимо переподключиться к сети (отключить и включить снова) и проверить доступность FreeIPA-сервера по FQDN-имени:

```
ping dc1.rosa.lan
```

4.4.2 Настройка клиента домена IPA

Перед установкой IPA-клиента необходимо настроить сеть. Для этого нужно выполнить следующие действия:

а) установить имя рабочей станции в домене, например:

```
sudo hostnamectl station.rosa.lan
```

б) задать для станции шлюз и DNS-сервер как IP-адрес IPA-сервера, используя интерфейс nmtui, а также указать домен поиска в rosa.lan;

в) перезагрузить ОС, чтобы изменения вступили в силу.

Если все сделано правильно, команда должна отрабатывать корректно, показывая соединение с IPA-сервером:

```
ping dc1
```

После перезагрузки и входа в Систему необходимо установить пакет IPA-клиента:

```
sudo dnf install ipa-client
```

Далее нужно запустить инсталляцию и вход в домен командой:

```
sudo ipa-client-install --mkhomedir
```

На этом этапе понадобится ввести пароль администратора домена. После перезагрузки можно входить доменным пользователем.

Для установки FreeIPA-клиента сначала нужно установить пакет ipa-client:

```
sudo dnf install ipa-client
```

Настроенный в соответствии с примером из п.4.4.1 IPA-сервер имеет хост и домен: dc1 / rosa.lan, соответственно клиент должен иметь такой же домен:

```
sudo hostnamectl set-hostname st1.rosa.lan
```

где:

– st1 – имя рабочей станции;
– rosa.lan – имя домена.

Следующим шагом необходимо ввести рабочую станцию в IPA-домен и инициализировать IPA-пользователя:

```
ipa-client-install –mkhomedir
```

На запросы инсталлятора следует дать следующие ответы:

– о перенастройке сервера времени chrony – NO;
– о продолжении установки с выбранными параметрами – YES;
– логин и пароль администратора IPA – admin и пароль администратора.

Если все прошло успешно, об этом будет сообщено в последней строке вывода на экран.

После этого следует войти на IPA-сервер и проверить присутствие рабочей станции в домене в списке "Узлы".

Затем нужно инициализировать IPA-пользователя, например, useripa1:

```
kinit useripa1
```

Далее нужно ввести пароль для пользователя FreeIPA-сервера – useripa1.

Затем необходимо задать новый пароль для пользователя вместо временного и подтвердить его.

4.4.3 Вход в домен

Для того чтобы пользователю войти в IPA-домен, необходимо на экране входа в ОС в момент выбора пользователя Системы выбрать параметр "Нет в списке", после чего откроется поле ввода логина нового пользователя.

Необходимо ввести логин с доменом, например, useripa1@ROSA.LAN (имя домена необходимо вводить заглавными буквами), и пароль, заданный в п.4.4.2 на предыдущем шаге.

После входа в рабочую станцию под доменным IPA-пользователем следует запросить его ID. ID пользователя должно совпадать с его ID на IPA-сервере.

На этом вход в домен IPA завершен.

5 СЕТЕВЫЕ ПОДКЛЮЧЕНИЯ

После установки ОС, при наличии активного проводного соединения, Система автоматически подключается к сети TCP/IP с помощью NetworkManager — программного обеспечения для управления сетевыми соединениями. Это происходит автоматически, если в сети имеется сервер DHCP, который назначает IP-адреса.

Для управления сетевыми подключениями в ОС используются несколько сервисов и утилит, которые позволяют произвести операции как через графический, так и консольный интерфейсы. Для работы с проводными соединениями широко используется NetworkManager. Если соединение с сетью обеспечивается с помощью беспроводного соединения, его можно легко настроить с помощью псевдографического интерфейса nmtui, также доступен консольный интерфейс nmcli.

5.1 Способы настройки сетевых соединений

5.1.1 NetworkManager

NetworkManager — это основной инструмент управления сетевыми подключениями в Системе, который представляет собой систему, управляющую сетевыми интерфейсами и соединениями, обеспечивающая их автоматическую настройку. NetworkManager поддерживает настройку как проводных, так и беспроводных соединений:

– при наличии подключенного сетевого кабеля NetworkManager автоматически подключает Систему к сети, используя настройки, предоставленные DHCP-сервером. Если требуется статическая настройка IP-адреса, это можно сделать через графический интерфейс или командную строку.

– при работе с Wi-Fi-сетями NetworkManager предоставляет возможность удобного выбора и подключения к доступным беспроводным сетям.

Настройка параметров NetworkManager осуществляется через графический интерфейс утилиты, который можно открыть несколькими способами:

– меню "Параметры Системы → Сеть и связь → Соединения";

– на нижней панели задач правой кнопкой мыши нужно нажать на пиктограмму сети → (рисунок 90) "Настроить сетевые соединения → Сеть и связь".

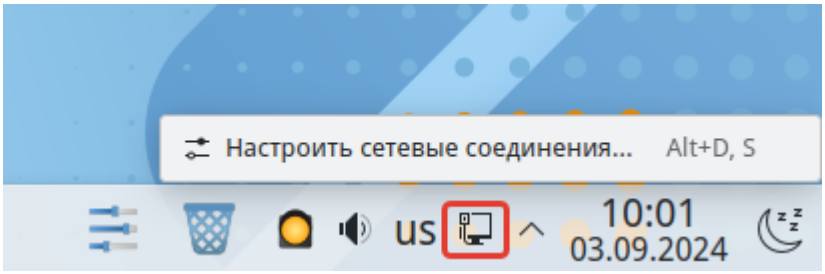


Рисунок 90 – Сетевые подключения

5.1.2 Использование nmtui

nmtui — это псевдографический интерфейс для управления сетевыми соединениями в текстовом режиме. Он предоставляет удобный способ настройки сетевых интерфейсов без необходимости работы с графическим интерфейсом.

Для запуска nmtui необходимо открыть терминал и выполнить команду

```
nmtui
```

После выполнения команды откроется текстовый интерфейс (рисунок 91), в котором можно выбрать нужное действие, например, подключение к сети или настройку нового соединения.

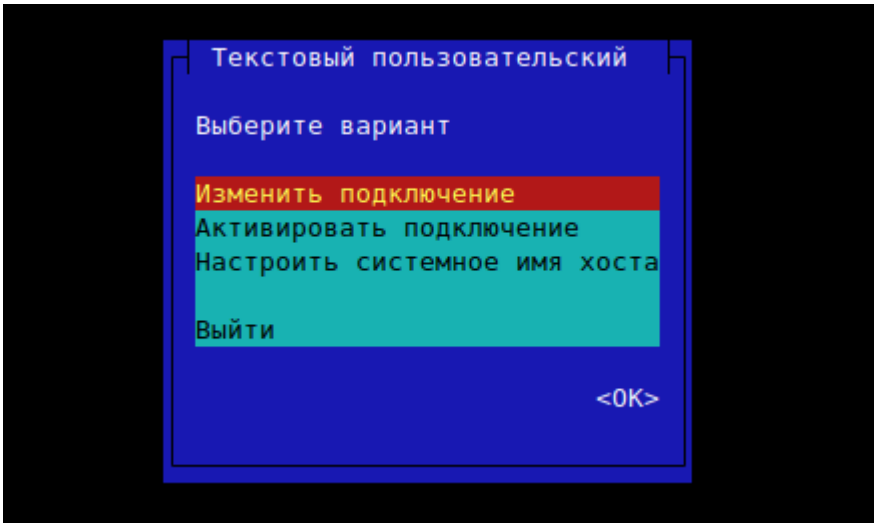


Рисунок 91 – Стартовый интерфейс утилиты nmtui

– Подключение к уже настроенным в Системе сетям осуществляется в разделе "Активировать соединение", где из списка можно выбрать доступное сетевое соединение и подключиться к нему.

– Создание нового соединения осуществляется в разделе "Изменить подключение", где можно настроить новое соединение, сначала выбрав необходимый тип соединения, затем указав параметры для проводного или беспроводного интерфейса (рисунок 92).

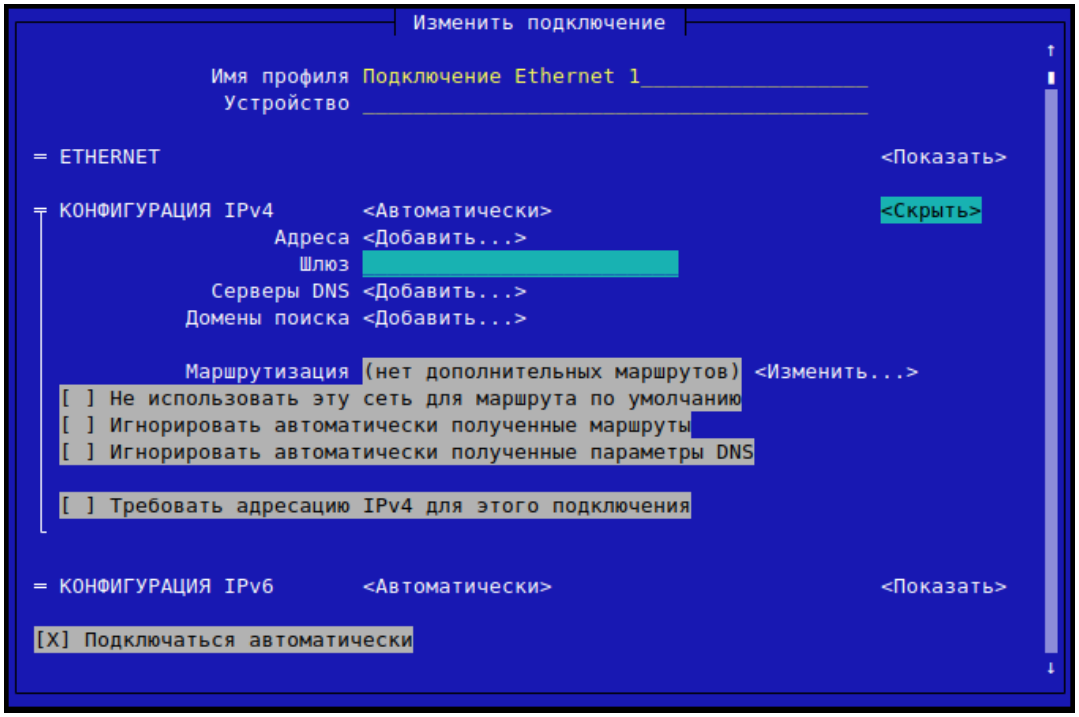


Рисунок 92 – Создание нового подключения в nmtui

– Для того, чтобы задать персонализированное имя хосту нужно выбрать соответствующий параметр "Настроить системное имя хоста" и далее в открывшимся окне задать новое имя (рисунок 93).

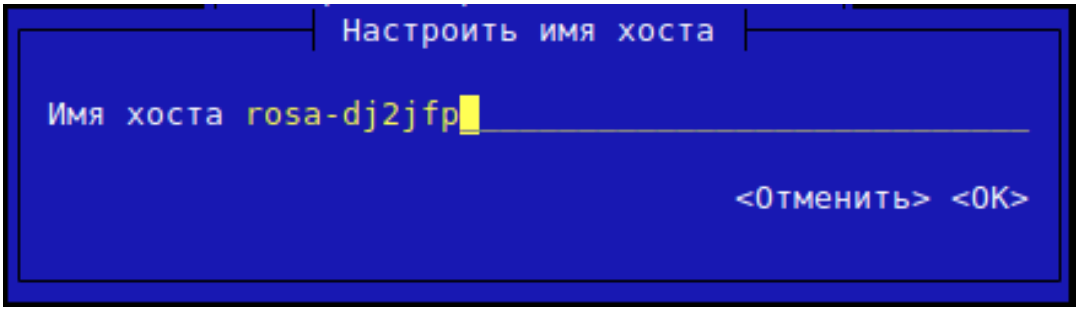


Рисунок 93 – Изменение имени хоста

5.1.3 Использование nmcli

`nmcli` – это утилита для управления NetworkManager из командной строки. Она предоставляет полный контроль над сетевыми интерфейсами и соединениями.

Для просмотра доступных сетевых интерфейсов и соединений нужно выполнить следующую команду:

```
nmcli device status
```

Для подключения к проводной сети нужно использовать команду:

```
nmcli con up id <имя_соединения>
```

Для подключения к беспроводной сети необходимо использовать команду:

```
nmcli device wifi connect <SSID> password <пароль>
```

Для создания нового соединения можно использовать следующую команду, указав все необходимые параметры:

```
nmcli con add
```

Для получения полной справки по использованию утилиты и ее параметрам можно использовать команду:

```
nmcli --help
```

5.1.4 Использование ifconfig и ip

Для отображения информации о сетевых интерфейсах и их текущем состоянии нужно использовать следующую команду:

```
ifconfig
```

Данная команда может использоваться и без прав root, но, если необходимо получить привилегии суперпользователя, нужно выполнить:

```
sudo ifconfig
```

Результатом выполнения команды станет получение списка имеющихся сетевых подключений и их параметров (рисунок 94).

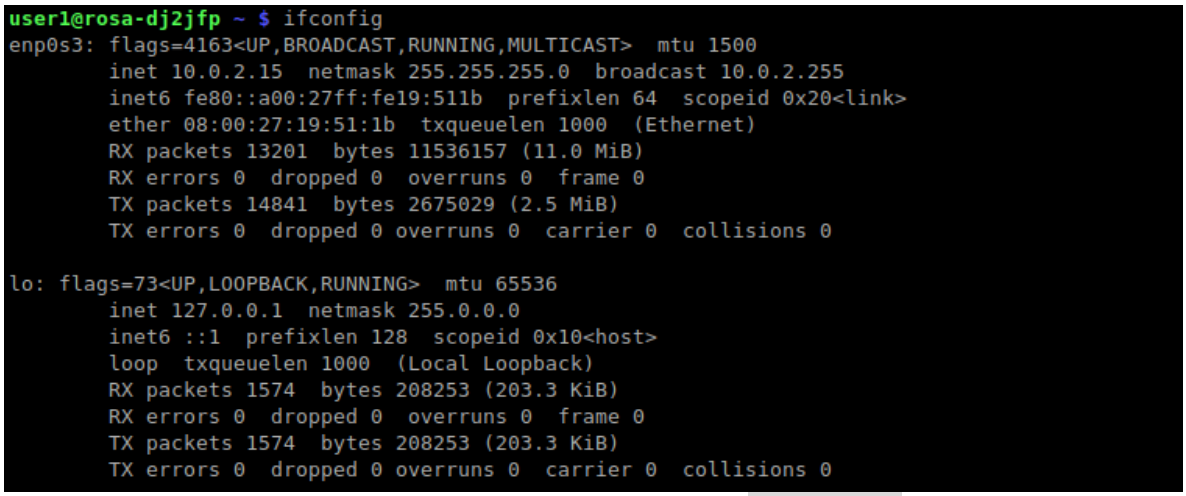


Рисунок 94 – Вывод команды `ifconfig`

Наряду с `ifconfig` рекомендуется использовать команду `ip`, которая предоставляет более гибкий и расширенный функционал по сравнению с `ifconfig`, что делает её предпочтительным инструментом для управления

сетевыми настройками в современных системах. Описание основных параметров и опций команды `ip` приведены ниже (таблица 10).

Таблица 10 - Описание параметров и опций команды `ip`

Наименование	Команда	Описание
Отображение информации о сетевых интерфейсах	`ip link show`	Отображает список всех сетевых интерфейсов на компьютере, включая их состояние и атрибуты
Отображение информации о сетевых адресах	`ip addr show`	Показывает IP-адреса и другие параметры сетевых интерфейсов, такие как маска подсети и статус
Назначение IP-адреса интерфейсу	`sudo ip addr add <IP-адрес>/<маска_подсети> dev <имя_интерфейса>`	Назначает указанный IP-адрес или маску подсети указанному сетевому интерфейсу
Удаление IP-адреса с интерфейса	`sudo ip addr del <IP-адрес>/<маска_подсети> dev <имя_интерфейса>`	Удаляет указанный IP-адрес или маску подсети с указанного сетевого интерфейса

Управление маршрутами	`ip route show`	Показывает текущие маршруты сети. Можно добавлять, изменять и удалять маршруты с помощью команд:
		`ip route add`
		`ip route change`
		`ip route del`

Управление VLAN	`sudo ip link add link <родительский_интерфейс> name <vlan_интерфейс> type vlan id <VLAN_ID>`	Создает виртуальный сетевой интерфейс VLAN на основе родительского интерфейса с указанным идентификатором VLAN
Просмотр IP-адреса сетевой карты	`ip a hostname -I`	Дает возможность просмотреть IP-адреса сетевой карты

Примечание – В таблице приведены лишь некоторые из основных возможностей команды `ip`. Команда `ip` также поддерживает множество других опций и параметров, которые могут быть использованы для более детальной настройки и управления сетевыми подключениями в ОС.

Чтобы получить полную информацию по команде `ip`, можно открыть справку с помощью команды:

```
man ip
```

5.1.5 Виртуальный сетевой интерфейс Loopback

Сетевой интерфейс Loopback (lo) представляет собой виртуальный сетевой интерфейс, который используется для тестирования сетевого стека Системы и для обеспечения локальной сетевой связности без необходимости подключения к физической сети.

В основном Loopback используется для тестирования сетевого стека ОС и отладки приложений, серверов, а также для работы локальных сервисов, таких как веб-серверы или базы данных. Поскольку пакеты, отправленные на адрес Loopback, не покидают компьютер, они полностью обрабатываются внутри сетевого стека. Это позволяет проверить, насколько правильно работает сетевая подсистема.

Использование интерфейса Loopback позволяет установить локальную сетевую связность без необходимости физического подключения к сети. Это полезно, например, для запуска локальных сетевых служб, которые должны быть доступны только локально.

Интерфейс Loopback также может использоваться для обеспечения безопасности и управления доступом. Некоторые службы и приложения могут быть настроены на прослушивание только на Loopback, чтобы они были доступны только локально и недоступны извне.

Основные характеристики Loopback интерфейса:

– интерфейс lo всегда присутствует в Системе и автоматически создается при старте ОС;

– стандартный IP-адрес, назначаемый этому интерфейсу — 127.0.0.1, который также известен как localhost. Этот адрес зарезервирован и используется исключительно для внутреннего общения Системы;

– обычно назначаемая маска подсети — 255.0.0.0, что указывает на широкую подсеть класса A.

Основные команды для работы с Loopback интерфейсом представлены ниже (таблица 11).

Таблица 11 - Описание параметров работы с интерфейсом Loopback

Наименование	Команда	Описание
Просмотр состояния интерфейса	`ip addr show lo`	Команда отображает текущее состояние интерфейса lo, включая его IP-адрес и маску подсети
Включение интерфейса	`sudo ip link set lo up`	Активирует интерфейс loopback, если он по каким-то причинам был отключен
Отключение интерфейса	`sudo ip link set lo down`	Отключает интерфейс lo, что крайне редко требуется, так как его отключение может нарушить работу локальных сервисов
Проверка соединения с localhost	`ping 127.0.0.1`	Команда `ping` позволяет убедиться, что loopback интерфейс функционирует нормально. В случае успеха появятся сообщения с откликами от 127.0.0.1

5.1.6 Сетевой интерфейс ens*

Сетевые интерфейсы, обозначаемые как ens*, являются стандартными именами для Ethernet-интерфейсов в современных Linux-системах. Эти имена заменили устаревшие обозначения типа eth0, eth1 и т.д. Структура имени ens* основывается на физическом расположении устройства в Системе, что делает его уникальным и независимым от порядка загрузки устройств.

Формат имени ens* обозначает Ethernet-интерфейс, где:

– en – указывает на тип подключения (Ethernet);
– s* — на номер слота или порта, к которому подключен интерфейс.

Интерфейсы ens* используются для проводных сетевых подключений. Они работают с различными типами сетей TCP/IP, обеспечивая взаимодействие компьютера с локальными сетями и Интернетом.

Для просмотра конфигурации всех ens* интерфейсов нужно использовать команду:

```
ip addr show ens*
```

В выводе команды будет представлена такая информация, как: IP-адрес подключений, состояние (включен или выключен), маска подсети и прочие параметры.

Для активации интерфейса ens* и приведения его в состояние, готовое для использования, нужно выполнить команду:

```
ip link set ens* up
```

Отключить интерфейс можно с помощью команды:

```
ip link set ens* down
```

Назначить статический IP-адрес интерфейсу ens* (маска /24 обозначает стандартную маску подсети 255.255.255.0 для IP-адреса 192.168.1.10) нужно с помощью команды:

```
sudo ip addr add 192.168.1.10/24 dev ens*
```

Проверка состояния соединения осуществляется командой:

```
ethtool ens*
```

В выводе команды будет выведена низкоуровневая информация об интерфейсе, включая состояние физического соединения, скорость линка и поддержку различных режимов работы.

В Системе интерфейсы ens* автоматически получают IP-адрес от DHCP-сервера при подключении, если не настроены на использование статического IP-адреса.

Для проверки текущего статуса DHCP или обновления IP-адреса можно использовать команду:

```
dhclient ens*
```

Эта команда запрашивает у DHCP-сервера новый IP-адрес для интерфейса ens*.

5.2 Настройка статических соединений

Для статической настройки всех соединений в сетевом интерфейсе можно воспользоваться одним из следующих способов:

– Через графический интерфейс в меню "Параметры Системы → Сеть и связь → Соединения" обычно заполняются такие параметры, как: "выбор DNS-сервера", "Домен поиска", "Метод настройки сети" (в данном случае – вручную), "IP-адрес", "Маска сети" и "Шлюз" (рисунок 95).

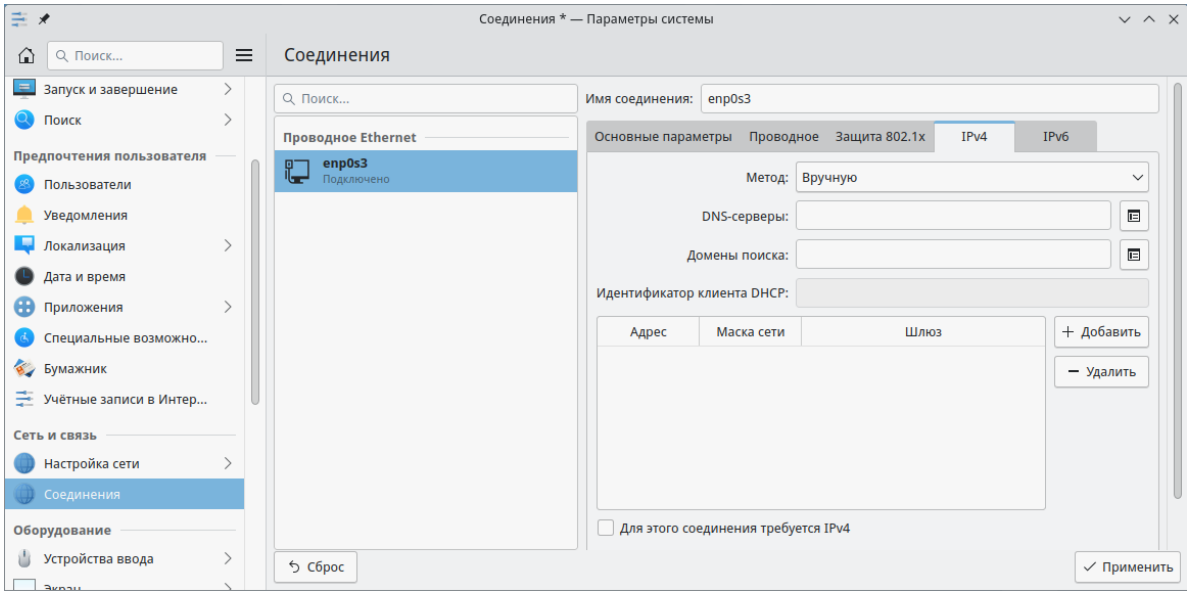


Рисунок 95 – Настройка сетевых соединений в графическом интерфейсе ОС

– для того чтобы прописать такие же параметры в консоли, необходимо выполнить команду со следующими параметрами:

```
sudo nmcli connection modify enp0s3 connection.autoconnect yes ipv4.method manual ipv4.dns 192.168.1.100 ipv4.dns-search rosa.lan ipv4.addresses 192.168.1.200/24 ipv4.gateway 192.168.1.1
```

где:

– connection modify – изменение соединения на интерфейсе enp0s3;

– connection.autoconnect yes – поднятие соединения при загрузке Системы;

– ipv4.method manual – задание соединения как статического;

– ipv4.dns – задание IP-адреса DNS-сервера;

– ipv4.dns-search – задание домена поиска;

– ipv4.addresses – задание IP-адреса и маски /24 интерфейса;

– ipv4.gateway – задание IP-адреса шлюза;

Затем нужно перезапустить интерфейс командами:

```
sudo nmcli connection down enp0s3
sudo nmcli connection up enp0s3
```

Можно использовать альтернативные команды для перезапуска:

```
nmcli device disconnect enp0s3
nmcli device connect enp0s3
```

5.2.1 Настройка статического IP-адреса

Для некоторых сетей может потребоваться использование статического IP-адреса. Это можно сделать как с помощью команды `nmtui` или `nmcli`.

– Использование `nmtui`: в открытом интерфейсе nmtui нужно выбрать "Изменить соединение", далее выбрать нужное соединение и установить статический IP-адрес, маску подсети и шлюз.

– Использование `nmcli`: для того, чтобы задать статический IP-адрес и шлюз для выбранного соединения нужно использовать команду:

```
nmcli con mod <имя_соединения> ipv4.addresses <IP/маска> ipv4.gateway <шлюз>
```

5.3 Настройка динамических соединений

Чтобы настроить все соединения динамически в сетевом интерфейсе, необходимо выполнить следующие действия:

а) в графическом интерфейсе ОС перейти в меню "Параметры системы → Сеть и связь → Соединения": обычно заполняются такие параметры, как "Метод настройки сети" (в данном случае – автоматически), "другие DNS-серверы", "Домены поиска", "Идентификатор клиента DHCP".

б) Для того чтобы прописать все параметры в консоли, нужно выполнить команду:

```
sudo nmcli connection modify enp0s3 ipv4.method auto ipv4.ignore-auto-dns no
```

где:

– connection modify – изменение соединения на enp0s3-интерфейсе;

– ipv4.ignore-auto-dns – задание параметра no (в таком случае не игнорируются полученные по DHCP DNS-серверы);

– ipv4.method auto – задание соединения динамическим.

Затем нужно перезапустить интерфейс командами:

```
sudo nmcli connection down enp0s3
sudo nmcli connection up enp0s3
```

5.3.1 Настройка динамических соединений (кроме DNS)

При необходимости можно настроить все соединения динамически в сетевом интерфейсе, за исключением DNS. Для этого следует придерживаться следующих действий:

а) в NetworkManager обычно заполняются такие параметры, как: "Метод настройки сети" (в данном случае – автоматически (только адреса)), "другие DNS-серверы", "Домены поиска", "Идентификатор клиента DHCP", "DNS";

б) для того чтобы прописать все параметры в консоли, нужно выполнить следующую команду:

```
sudo nmcli connection modify enp0s3 ipv4.ignore-auto-dns yes ipv4.dns 192.168.1.100
```

где:

– connection modify – изменение соединения на enp0s3-интерфейсе;

– ipv4.ignore-auto-dns – задание параметра yes (в таком случае игнорируются полученные по DHCP серверы DNS);

– ipv4.dns – задание IP-адреса пользовательского DNS-сервера;

в) далее нужно перезапустить интерфейс с помощью команд:

```
sudo nmcli connection down enp0s3
sudo nmcli connection up enp0s3
```

5.4 Переадресация

Переадресация сетевых пакетов (форвардинг) – это процесс передачи сетевых пакетов от одного сетевого интерфейса к другому, что позволяет Системе функционировать как маршрутизатор. Включение переадресации в Системе может быть полезно для настройки сетевых шлюзов, маршрутизаторов или при организации общего доступа к интернет-соединению.

Переадресация IP отключена в Системе по умолчанию, и для ее включения требуется наличие дополнительного файла. Если такого файла нет, нужно его создать. Имя файла состоит из имени сетевого интерфейса, за которым следует суффикс .network, например, tun0.network для сетевого интерфейса /dev/tun0. При этом другие расширения игнорируются.

Далее в созданном файле необходимо внести изменения. Для примера можно рассмотреть настройку сетевого интерфейса /dev/tun0 в файле tun0.network. Он состоит из двух разделов: "Match" и "Network". В разделе "Match" нужно указать имя сетевого интерфейса, а в разделе "Network" включить "IP Forwarding":

```
# cat /etc/systemd/network/tun0.network
[ Match ]
Name = tun0
[ Network ]
```

```
IPForward = ipv4
```

Чтобы проверить текущее значение функции net.ipv4.ip_forward нужно выполнить команду:

```
cat /proc/sys/net/ipv4/ip_forward
```

Если команда возвращает значение "0", значит функция IPForward выключена, если "1" – включена.

Также можно проверить значение через команду `sysctl`:

```
sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

5.4.1 Включение переадресации IPv4

Для включения переадресации IPv4 необходимо изменить соответствующие настройки в системных параметрах. Основной параметр, отвечающий за переадресацию, находится в /proc/sys/net/ipv4/ip_forward. Чтобы включить переадресацию, нужно выполнить следующую команду:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Эта команда активирует переадресацию пакетов в Системе до момента ее перезагрузки, позволяя Системе пересылать их между сетевыми интерфейсами.

Чтобы сделать это изменение постоянным (переживающим перезагрузку Системы), необходимо добавить или изменить строку в файле /etc/sysctl.conf:

```
grep forward /etc/sysctl.conf
net.ipv4.ip_forward = 1
```

Затем нужно применить изменения с помощью команды:

```
sysctl -p
```

5.4.2 Включение переадресации IPv6

Аналогично можно включить переадресацию для IPv6. Для этого следует использовать команду:

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Чтобы сделать изменение постоянным, нужно добавить строку в файл /etc/sysctl.conf:

```
net.ipv6.conf.all.forwarding = 1
```

5.4.3 Настройка правил переадресации с использованием nftables

С помощью nftables можно создать правила, управляющие перенаправлением пакетов между сетевыми интерфейсами.

Пример настройки NAT (Network Address Translation) для переадресации пакетов с внутреннего интерфейса ens33 на внешний ens34:

```
    nft add table ip nat
    nft add chain ip nat postrouting { type nat hook postrouting
priority 100 \; }
    nft add rule ip nat postrouting oifname "ens34" masquerade
    nft add chain ip nat prerouting { type nat hook prerouting
priority 0 \; }
    nft add chain ip nat input { type nat hook input priority 0
\; }
    nft add chain ip nat output { type nat hook output priority
100 \; }
    nft add chain ip nat forward { type nat hook forward priority
0 \; }
```

Эти команды создают таблицу и правила, которые позволяют устройствам из внутренней сети (подключенным через ens33) выходить в интернет через интерфейс ens34.

После настройки переадресации можно использовать команду для проверки активных правил и диагностирования возможных проблем:

```
    nft list ruleset
```

Для мониторинга сетевого трафика и подтверждения работы переадресации можно использовать утилиты, такие как tcpdump.

5.4.3.1 Настройка динамических соединений (кроме DNS)

При необходимости настроить все соединения динамически в сетевом интерфейсе, за исключением DNS, следует придерживаться следующих действий:

а) В графической оболочке NetworkManager обычно заполняются такие параметры, как: "Метод настройки сети" (в данном случае – автоматически (только адреса)), "другие DNS-серверы", "Домены поиска", "Идентификатор клиента DHCP, DNS";

б) Для того чтобы прописать все параметры в консоли, нужно выполнить следующую команду:

```
sudo nmcli connection modify enp0s3 ipv4.ignore-auto-dns yes
ipv4.dns 192.168.1.100
```

где:

– connection modify – изменение соединения на enp0s3-интерфейсе;

– ipv4.ignore-auto-dns – задание параметра yes (в таком случае игнорируются полученные по DHCP DNS-серверы);

– ipv4.dns – прописываем IP-адрес нашего DNS-сервера;

в) далее нужно перезапустить интерфейс командами:

```
sudo nmcli connection down enp0s3
sudo nmcli connection up enp0s3
```

5.5 Мониторинг и диагностика сети

NetworkManager и утилиты командной строки также предоставляют инструменты для мониторинга состояния сети и диагностики проблем.

Для отображения текущего состояния NetworkManager нужно выполнить команду:

```
nmcli general status
```

Для просмотра детальной информации о сетевых интерфейсах нужно использовать команду:

```
nmcli device show
```

Для диагностики проблем с сетевыми соединениями необходимо использовать стандартные утилиты `ping` и `traceroute`.

Утилита `ping` предназначена для проверки доступности узлов в сети и измерения времени отклика между ними. Она отправляет ICMP-пакеты на указанный адрес и ожидает ответа. Использование команды:

```
ping <адрес_хоста>
```

Команда отображает результаты в виде времени отклика и процента потерь пакетов, что помогает определить, доступен ли узел и имеются ли сетевые проблемы, такие как высокая задержка или потеря пакетов.

Утилита `traceroute` используется для диагностики маршрута, по которому проходят пакеты до указанного узла. Она выводит список промежуточных узлов (роутеров), через которые проходят пакеты. Это помогает выявить, на каком этапе возникают сетевые проблемы.

Команда для выполнения трассировки:

```
traceroute <адрес_хоста>
```

Вывод показывает последовательность узлов и время, затраченное на прохождение пакетов через каждый из них, что позволяет идентифицировать узкие места или проблемы в сети.

5.6 Межсетевой экран Nftables

5.6.1 Общие сведения о Nftables

Стандартным сетевым экран в ОС принят стандарт использования Nftables, который заменил собой использование iptables.

Nftables – это более простая и эффективная альтернатива iptables/ip6tables/arptables. Она предлагает унифицированную обработку IPv4 и IPv6, а также улучшенную производительность и поддержку современных сетевых протоколов. Для nftables была разработана новая система фильтрации пакетов, добавлена пользовательская утилита ntf, а также создан слой совместимости с iptables/ip6tables. Файервол `nftables` использует набор хуков, систему отслеживания соединений, систему очередей и подсистему логирования netfilter.

Как и iptables, nftables построен на правилах, определяющих действия. Эти правила содержаться в таблицах (table), к которым прикреплены цепочки (chain). Цепочка может содержать набор правил и исполняется в хуках (hook) netfilter. Таблица специфична для каждого из протоколов (ARP/IPv4/IPv6). Одно из основных отличий от iptables заключается в том, что больше нет предопределенных таблиц и цепочек.

Описание правил nftables:

– Таблицы (**table**):

  – table – это контейнер для цепочек (chain).

– Семейства таблиц (**family**):

  – ip – цепочки, связанные с IPv4;
  – ip6 – цепочки, связанные с IPv6;
  – arp – цепочки, связанные с ARP;
  – bridge – соединение связанных цепочек;
  – inet – смешанные цепочки ipv4 / ipv6;
  – netdev – цепочки, которые фильтруют в начале стека.

– Цепочки (**chain**):

  – chain – группировка правил;

– base chain – цепочка с зарегистрированным хуком netfilter, типом и приоритетом;

– other chain – цепочка без привязки к хуку и по умолчанию без отслеживания трафика.

Существует три вида цепочек:

– filter – стандартный тип; фильтр пакетов;

– nat – протоколы IP/IPv6/INET; хуки PREROUTING, INPUT, OUTPUT, POSTROUTING; преобразование NAT;

– route – протоколы IP/IPv6/; хуки OUTPUT; маршрутизация.

– Хук (**hook**) – это представление (правило) от том, на каком этапе происходит обработка:

– prerouting – до принятия решения о маршрутизации; для пакетов, входящих в машину;

– input – для всех пакетов, входящих в локальную Систему;

– forward – для пересылаемых пакетов вне локальной Системы;

– output – для пакетов, исходящих из локальной Системы;

– postrouting – после принятия решения о маршрутизации; для пакетов, покидающих машину;

– priority – число, используемое для упорядочивания цепочек или для установки приоритетов между некоторыми операциями;

– policy – для пакетов, которые явно не приняты или отклонены в содержащихся правилах.

– Правила (**rules**):

– rule – определяет, какое действие необходимо выполнить для каких пакетов; правила прикреплены к цепочкам; каждое правило может иметь выражение для сопоставления пакетов и одно или несколько действий, выполняемых при сопоставлении;

– statements – представляет собой действие, которое должно быть выполнено при совпадении правила:

– accept – прием пакета и прекращение выполнения текущего правила;

– drop – отказ от приема пакета и прекращение выполнения текущего правила;

– reject – отклонение пакета с сообщением ICMP;

– queue – постановка пакета в очередь в пользовательское пространство и прекращение выполнения текущего правила;

– continue – продолжение выполнения текущего и переход к следующему правилу;

– return – возврат из текущей цепочки и переход к следующему правилу последней цепочки; в базовой цепочке это эквивалентно accept;

– jump – продолжение с первого правила; будет продолжено в следующем правиле после выдачи оператора возврата;

– goto – аналогично jump, но после новой цепочки оценка продолжится с последней цепочки вместо той, которая содержит оператор goto;

– handle – внутренний номер, идентифицирующий определенное правило;

– position – внутренний номер, используемый для вставки правил до определенного handle.

5.6.2 Установка и запуск Nftables

В ОС по умолчанию уже установлен фреймворк nftables.

Если по каким-то причинам пакет подсистемы удален, nftables можно установить следующей командой:

```
sudo dnf install nftables
```

Запуск сервиса nftables осуществляется командой:

```
sudo systemctl start ntfables.service
```

Чтобы включить запуск при старте Системы, нужно воспользоваться командой:

```
sudo systemctl enable nftables.service
```

5.6.3 Работа с Nftables

В качестве базового примера работы с nftables нужно выполнить следующие шаги для закрытия/открытия порта:

а) создать таблицу с помощью команды:

```
sudo /usr/sbin/nft add table inet my_table
```

Формат команды:

```
nft add table <family> <name>
```

где:

– <family> – один из протоколов ip/ip6/inet/arp/bridge/netdev;
– <name> – имя новой таблицы.

б) проверить наличие новой таблицы:

```
sudo nft -a list ruleset
```

в) добавить цепочку:

```
sudo /usr/sbin/nft add chain inet my_table input { type filter hook input priority 0 \; policy accept \; }
```

Формат команды:

```
nft add chain <family> <name table> <name chain> {hook}
```

где:

– <family> – один из протоколов ip/ip6/inet/arp/bridge/netdev;
– <name table> – имя таблицы;
– <name chain> – имя новой цепочки.

г) проверить наличие новой цепочки:

```
sudo /usr/sbin/nft -a list ruleset
```

д) добавить правило, с помощью которого ограничивается доступ к порту 80:

```
sudo /usr/sbin nft add rule inet my_table input tcp dport 80 drop
```

Формат правила:

```
nft add rule <family> <name table> <protocol> <dport> <port> <event>
```

где:

– <family> – один из протоколов nftable ip/ip6/inet/arp/bridge/netdev;
– <name table> – имя таблицы;
– <name chain> – имя новой цепочки;
– <protocol> – протокол TCP/UDP/ICMP и т.д.;
– <dport> – входящий порт (destination port);
– <port> – номер входящего порта;
– <event> – событие drop/accept/jump и т.д.

е) для открытия порта 80 нужно выполнить:

```
sudo /usr/sbin nft add rule inet my_table input tcp dport 80 accept
```

Но получить доступ к порту 80 после этого не удастся, так как в списке правил сначала все пакеты на порт 80 блокируются, а потом "пропускаются". В рассматриваемом случае пропускать уже нечего, так как изначально на этом порту всё блокируется.

В этом случае нужно удалить запрещающее правило с drop, для чего с помощью отметки "handle 2", которая присутствует в выводе на экран, выполнить команду:

```
sudo /usr/bin/nft delete rule inet my_table input handle 2
```

Далее следует проверить результат:

```
sudo /usr/sbin/nft -a list ruleset
```

Теперь доступ к порту 80 есть в полном объеме.

Если необходимо полностью очистить таблицу, то следует выполнить команду:

```
sudo /usr/sbin/nft delete table inet my_table
```

Если требуется полностью очистить файервол (все цепочки и таблицы), то следует выполнить:

```
sudo /usr/sbin/nft flush ruleset
```

5.7 Синхронизация времени с использованием Chrony

Chrony — это служба синхронизации времени, предназначенная для точного поддержания системного времени. В отличие от классического NTP-демона, Chrony обеспечивает более высокую точность в условиях нестабильных сетей, при больших задержках, а также на системах с нерегулярной загрузкой (например, на виртуальных машинах).

Chrony состоит из двух основных компонентов:

– chronyd — демон, выполняющий синхронизацию времени;
– chronyc — консольная утилита для управления демоном и получения статистики.

Основной файл конфигурации размещается по адресу /etc/chrony.conf.

В базовой конфигурации определяются:

– список внешних NTP-серверов:

```
server ntp.server.example iburst
```

– файл дрейфа часов (погрешности локального кварца):

```
driftfile /var/lib/chrony/drift
```

– параметры доступа клиентов (при использовании Chrony как сервера):

```
allow 192.168.0.0/24
```

После внесения изменений необходимо перезапустить службу:

```
systemctl restart chronyd
```

Проверка, что служба запущена и работает корректно:

```
systemctl status chronyd
```

Служба chronyd отвечает за работу Chrony как клиента или сервера синхронизации времени. Управление службой осуществляется стандартными инструментами systemd:

– запуск и остановка службы:

```
systemctl start chronyd
systemctl stop chronyd
```

– включение автозапуска:

```
systemctl enable chronyd
```

– проверка статуса:

```
systemctl status chronyd
```

– Перезагрузка службы для применения изменений конфигурации:

```
systemctl restart chronyd
```

Для проверки состояния и источников времени Chrony предоставляет две основные утилиты: chronyc tracking и chronyc sources.

Для просмотра текущего статуса синхронизации, который показывает, насколько локальное время синхронизировано с источником, используется:

```
chronyc tracking
```

Основные параметры утилиты:

– Reference ID / Name — с каким сервером синхронизируется система;
– Stratum — уровень источника времени;
– Last offset — последняя измеренная временная разница;
– Frequency — корректировка частоты локального генератора;
– Leap status — корректность времени (normal / insert / delete).

Для вывода всех доступных серверов времени используется:

```
chronyc sources -v
```

Статусы источников указываются в выводе в колонке MS:

– * — главный используемый источник (selected);
– + — кандидат на использование;
– ? — источник недоступен;
– x — источник отклонён как неверный;

– ~ — источник нестабилен.

Проверка смещения времени:

```
chronyc sourcestats
```

Проверка корректности локального времени (используется для принудительного мгновенного исправления времени):

```
chronyc makestep
```

5.7.1 Настройка Chrony в качестве сервера времени

В случае необходимости центральной синхронизации времени для группы рабочих станций Chrony может быть использован как локальный NTP-сервер. Это позволяет обеспечить единообразие времени в закрытом контуре или при отсутствии доступа к внешним серверам.

Чтобы предоставить другим узлам возможность синхронизироваться с данной машиной, в файле конфигурации /etc/chrony.conf требуется добавить строку:

```
allow <подсеть>
```

Например, для разрешения доступа внутренней сети 192.168.1.0/24:

```
allow 192.168.1.0/24
```

Необходимо убедиться, что служба chronyd запущена:

```
systemctl restart chronyd
```

Если сервер должен быть эталонным источником при отсутствии доступа к Интернету или NTP-сетям, используется директива:

```
local stratum 10
```

Директива указывает, что сервер предоставляет локальное время с определённым (не нулевым) stratum.

Для проверки доступности сервера со стороны клиента на клиентских системах необходимо выполнить команду:

```
chronyc sources
```

Сервер должен отображаться в списке источников времени со статусом "*" или "+".

6 НАСТРОЙКА И УПРАВЛЕНИЕ СЕТЕВЫМИ СЕРВИСАМИ

6.1 Сервер DHCP

Сервер DHCP (Dynamic Host Configuration Protocol) используется для автоматического назначения IP-адресов и других сетевых параметров (например, шлюз по умолчанию, DNS-серверы) устройствам в сети. Это позволяет упростить процесс подключения устройств к сети, так как нет необходимости вручную настраивать параметры сетевого интерфейса для каждого устройства.

Когда устройство (клиент) подключается к сети, оно отправляет запрос на сервер DHCP. Сервер DHCP, в свою очередь, выделяет свободный IP-адрес из заранее настроенного диапазона и отправляет его клиенту, а также другие параметры конфигурации сети.

6.1.1 Установка

Установка DHCP выполняется командой:

```
sudo dnf -y install dhcp
```

Следует обратить внимание, что обычно к Системе подключено более одного сетевого интерфейса. Если нужно, чтобы DHCP-сервер был запущен лишь на одном из интерфейсов, то необходимо выполнить следующие действия:

а) Открыть файл /etc/sysconfig/dhcpd;

б) Добавить имя конкретного интерфейса в список DHCPDARGS, например, если интерфейс `eth0`, то добавить:

```
DHCPDARGS=eth0
```

в) Сохранить этот файл и выйти.

6.1.2 Настройка

Первым шагом установки DHCP-сервера является создание файла конфигурации dhcpd.conf. Основной файл конфигурации DHCP обычно располагается по пути /etc/dhcp/dhcpd.conf (изначально пуст) и хранит всю сетевую информацию, отправляемую клиентам.

Тем не менее, существует другой пример файла конфигурации /usr/share/doc/dhcp*/dhcpd.conf.sample, который является хорошей отправной точкой для настройки DHCP-сервера.

В файле конфигурации DHCP существует два типа операторов:

– parameters (параметры) – определяет, как выполнять задачу, выполнять ли задачу вообще и какие параметры конфигурации сети отправлять DHCP-клиенту;

– declarations (объявление) – определяет топологию сети, клиентов, предлагает адреса для клиентов и применяет группу параметров к группе объявлений.

Для настройки DHCP-сервера выполняют следующие действия:

а) скопировать пример файла конфигурации в качестве основного:

```
cp /usr/share/doc/dhcp-4.2.5/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

б) открыть для редактирования основной файл конфигурации:

```
sudo nano /etc/dhcp/dhcpd.conf
```

в) задать глобальные параметры DHCP-сервера, которые будут применяться ко всем подсетям:

```
option domain-name "rosa.lan";
option domain-name-servers ns1.rosa.lan, ns2.rosa.lan;
default-lease-time 3600;
max-lease-time 7200;
authoritative;
```

г) определить подсеть (в этом примере сделана настройка DHCP для 192.168.56.0/24 LAN-сети):

```
subnet 192.168.56.0 netmask 255.255.255.0 {
option routers 192.168.56.1;
option subnet-mask 255.255.255.0;
option domain-search "rosa.lan";
option domain-name-servers 192.168.56.1;
range 192.168.56.10 192.168.56.100;
}
```

Все строки параметров в файле конфигурации заканчиваются точкой с запятой (;). Некоторые параметры могут иметь несколько значений. Например, domain-name-servers, у которого два IP-адреса, разделенных запятой. Строки, начинающиеся со знака решетки (#), являются комментариями и не обрабатываются DHCP-сервером.

Некоторые общие параметры DHCP-сервера:

– subnet – объявляет подсеть (в вышеприведенном случае 192.168.38.0 с маской 255.255.255.0);

– range – диапазон выдаваемых адресов (от 192.168.56.10 до 192.168.56.100);

– option subnet-mask – маска сети (255.255.255.0);

– option broadcast-address – широковещательный адрес (192.168.38.255);

– domain-name-servers – адреса серверов DNS (8.8.8.8, 8.8.4.4);

– option domain-name – доменное имя (workgroup);

– option routers – IP-адрес шлюза или точки выхода в сеть (192.168.56.1).

д) После внесения изменений в основной файл конфигурации нужно открыть файл /etc/default/isc-dhcp-server и заменить параметр INTERFACESv4 на имя сетевого интерфейса, который смотрит внутрь сети. Чтобы узнать его имя, следует воспользоваться командами `ipconfig` или `ip`.

6.1.3 Присвоение адреса

Для того чтобы присвоить статический IP-адрес конкретному клиентскому компьютеру в сети, нужно определить раздел в файле /etc/dhcp/dhcpd.conf, в котором необходимо указать MAC-адреса и фиксированный IP-адрес для назначения, например:

```
host rosa {
hardware ethernet 00:f0:m4:6y:89:0g;
fixed-address 192.168.56.105;
}
host rv-node {
hardware ethernet 00:4g:8h:13:8h:3a;
fixed-address 192.168.56.110;
}
```

Примечание – узнать MAC-адрес в ОС РОСА "ХРОМ" можно, используя следующую команду:

```
ifconfig -a eth0 | grep HWaddr
```

После этого следует запустить DHCP-службу и включить параметр для её автоматического запуска со следующей загрузкой Системы, используя команды:

```
sudo systemctl start dhcpd
sudo systemctl enable dhcpd
```

Затем необходимо разрешить DHCP-службу (DHCPD-служба прослушивает порт 67/UDP) с помощью команды:

```
nft add rule inet filter input udp dport 67 accept
```

Также разрешить DHCP-службу можно с помощью следующих команд:

```
    nft add rule inet filter input tcp dport 67 ct state new
accept
    nft list ruleset > /etc/nftables.conf
```

Затем можно загрузить эти правила при старте:

```
    nft -f /etc/nftables.conf
```

6.1.4 Настройка клиента

Далее необходимо настроить клиентов в сети на автоматическое получение IP-адресов от DHCP-сервера. Для этого нужно войти на клиентский компьютер и перейти к редактированию файла конфигурации интерфейса Ethernet командой (имя/номер интерфейса приведены в качестве примера):

```
    sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

В файл необходимо добавить следующие строки:

```
    DEVICE=eth0
    BOOTPROTO=dhcp
    TYPE=Ethernet
    ONBOOT=yes
```

Затем следует сохранить файл и выйти из редактора.

6.2 Сервер DNS

DNS является основным компонентом сетевой инфраструктуры, обеспечивающим преобразование доменных имен в IP-адреса и обратно. DNS является критическим элементом сетевой инфраструктуры, и его правильная настройка и управление играют важную роль в обеспечении работоспособности сетей.

В качестве сервера, реализующего DNS, в ОС используется сервер Bind.

6.2.1 Установка

Перед установкой сервера необходимо обновить Систему:

```
    dnf --refresh distro-sync
```

Далее нужно перейти к установке сервера Bind, исполняемый файл которого называется named. **Следует обратить внимание**, что все

конфигурационные файлы/сервисы будут содержать в названии префикс named*.

Далее для настройки DNS-сервера необходимо настроить параметр systemdresolved в файле /etc/systmd/resolved.conf.

Перед настройкой также следует запросить команду:

```
lsof -i :53
```

Исправить нужно следующие строки:

```
DNS=127.0.0.1
FallbackDNS=
DNSSEC=no
LLMNR=resolve
DNSStubListener=no
```

где:

– DNS=127.0.0.1 – задается локальный IP-адрес, на котором будет работать будущий DNS-сервер;

– FallbackDNS= – данный параметр нужно оставить пустым, чтобы systemd-resolved не переключался на fallback DNS-сервера;

– DNSSEC=no – отключается DNSSEC;

– LLMNR=resolve – LLMNR переводится в режим resolve;

– DNSStubListener=no – задается параметр no, чтобы systemd-resolved не прослушивал порт 53.

Остальные опции в этом файле не следует менять. Далее нужно перезапустить systemd-resolved:

```
sudo systemctl restart systemd-resolved
```

После запуска вновь требуется запросить команду (вывод команды должен быть пустой):

```
lsof -i :53
```

Если вывод действительно пустой, значит всё сделано корректно и можно приступать к настройке и запуску bind.

6.2.2 Настройка

Основной конфигурационный файл сервиса bind – named.conf.

Для настройки нужно выполнить следующие действия:

а) Открыть конфигурационный файл named.conf и для минимальной работы DNS-сервера исправить следующие строки:

```
    options {
    listen-on port 53 { any; }; listen-on-v6 port 53 { any; }; .
. . . .
    allow-query { any; };
```

где:

– listen-on port 53 { any; }; – задается параметр any для прослушивания на всех IPv4 хоста;

– listen-on-v6 port 53 { any; }; – задается параметр any для прослушивания на всех IPv6 хоста;

– allow-query { any; }; – разрешаются любые запросы к серверу.

б) Запустить bind:

```
    sudo systemctl start named.service
```

в) Проверить, какие службы прослушивают порт 53:

```
    lsof -i :53
```

Если в выводе прописан параметр named, то все настроено правильно.

г) Проверить работу DNS-сервера с помощью команды, например, с адресом yandex.ru:

```
    dig @127.0.0.1 yandex.ru
```

Также для проверки можно использовать следующую команду:

```
    nslookup yandex.ru
```

Должны быть выведены IP-адреса yandex.ru.

д) Включить DNS-сервис в автозагрузку:

```
    sudo systemctl enable named.service
```

6.2.3 Переадресация

Чтобы настроить bind для переадресации (forward) запросов к другим DNS-серверам нужно выполнить следующие шаги:

а) Открыть конфигурационный файл named.conf и для минимальной работы DNS-сервера исправить следующие строки:

```
    options {
    . . . . .
    recursion yes; . . . . .
    forward only; forwarders {
    77.88.8.8;
```

```
77.88.8.1;
};
```

где:

– forward – режим перенаправления:

– forward only – все запросы на настраиваемый DNS-сервер будут перенаправляться на другие DNS-сервера, прописанные в следующей опции forwarders {};

– forward first – все запросы на настраиваемый DNS-сервер будут перенаправляться на другие DNS-сервера, прописанные в следующей опции forwarders {}. Если с помощью них не удастся разрешить запрос, то запрос будет пытаться разрешаться настраиваемым DNS-сервером локально;

– forwarders { 77.88.8.8; 77.88.8.1; } – список DNS-серверов, для перенаправления запросов.

б) Запустить DNS-сервер:

```
sudo systemctl start named.service
```

В результате DNS-сервер будет настроен для минимальной работы.

6.3 Сервер Samba

Samba представляет собой программный пакет с открытым исходным кодом, дающий сетевым администраторам возможность гибко и свободно настраивать, конфигурировать и выбирать системы и оборудование, то есть устанавливать на компьютерах с UNIX/Linux-подобными ОС имитации устройств с Windows. Это упрощает выполнение задач по обмену файлами (как файл-серверам) или задавать параметры печати в качестве принт-серверов.

Samba обеспечивает свободный доступ к:

– дискам Linux – к Windows-компьютерам;
– дискам Windows – к оборудованию с ПО Linux;
– принтерам Linux – к Windows-компьютерам;
– принтерам Windows – к Linux-системам.

6.3.1 Настройка сети

Для обеспечения работы Samba используются протоколы SMB и CIFS.

В отличие от SMB, CIFS имеет более широкий функционал – этот протокол способен поддерживать символические и жесткие ссылки, а также файлы больших размеров.

Но в настоящее время CIFS почти не используется, так как в последних версиях Windows (начиная с версии 8) используются протоколы SMB2 или SMB3, которые по сути являются обновленной версией CIFS.

В качестве примера ниже описывается создание лабораторной инфраструктуры, состоящей из одного сервера и одного клиента, находящихся в одной подсети 192.168.122.0/24. Все приведенные консольные команды выполняются от пользователя root, если не указано иное.

Важно правильно настроить сеть на клиенте и сервере. В простых случаях, в том числе при поднятии тестовых стендов, необходимо, чтобы клиент и сервер находились в одной подсети. Например, чтобы у ВМ с сервером был IP-адрес 192.168.10.2, а у ВМ с клиентом – 192.168.10.3.

Для настройки сети на сервере необходимо выполнить следующие шаги:

а) Установить полное доменное имя (FQDN) в формате <имя_хоста>.<домен>.<зона>, например, samba1.rosa.lan.

В этом случае именем хоста будет samba1, доменом – rosa, доменной зоной – lan, realm – rosa.lan.

Следует избегать использования зоны .local. Нужно или выключить ее, или перенастроить Avahi, чтобы использовать ее.

Для установки имени хоста на контроллере домена используется команда:

```
sudo hostnamectl set-hostname samba1.rosa.lan
```

Посмотреть IP-адрес можно с помощью команды:

```
/sbin/ip a
```

Пример вывода:

```
[root@server1 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet
127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever inet6 ::1/128 scope
host
valid_lft forever preferred_lft forever
2: host0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue state UP group default qlen 1000
link/ether 32:e2:0b:ba:17:c5 brd ff:ff:ff:ff:ff:ff link-
netnsid 0
```

```
    inet 169.254.212.10/16 brd 169.254.255.255 scope link host0\
valid_lft forever preferred_lft forever
    inet 192.168.122.12/24 brd 192.168.122.255 scope global\
dynamic
    host0
    valid_lft 2577sec preferred_lft 2577sec inet6\
fe80::30e2:bff:feba:17c5/64 scope link valid_lft forever\
preferred_lft forever
```

В данном примере IPv4-адрес – 192.168.122.12.

б) Добавить в файл /etc/hosts строку такого вида, чтобы и FQDN, и краткое имя выводились в IP-адрес контроллера домена:

```
    192.168.122.12 samba1.rosa.lan samba1
```

в) Проверить корректность записей в файле /etc/hosts можно командой:

```
    ping -c3 samba1.rosa.lan ping -c3 samba1
```

Следует обратить внимание, что сеть должна запускаться до входа пользователя в Систему. Если используется подключение к сети по Wi-Fi, то необходимо настроить его соответствующим образом.

6.3.2 Запуск контроллера домена

Контроллер домена – сервер, управляющий доступом к сетевым ресурсам в рамках одного домена (группы хостов или сетей, которые объединены общими политиками безопасности).

Для запуска контроллера домена нужно выполнить следующие шаги:

а) Установить необходимые для работы пакеты:

```
    sudo dnf install samba-server /bin/ps /usr/bin/xargs\
/usr/bin/nslookup
```

б) Отключить лишние службы из состава набора программ Samba, так как их обособленная работа не требуется и будет мешать контроллеру домена:

```
    sudo systemctl disable --now smb nmb winbind
```

в) Убедиться, что они действительно выключены, используя команду, вывод которой должен быть пустым:

```
    ps ax | grep -E "samba|smbd|nmbd|winbindd" | grep -v grep
```

г) Удалить старые файлы настроек, оставив их резервные копии:

```
    mv -v /etc/samba/smb.conf /etc/samba/smb.conf.old mv -v\
/etc/krb5.conf /etc/krb5.conf.old
```

д) Определить старые БД, путь к которым можно узнать командой:

```
    smbd -b | grep -E "LOCKDIR|STATEDIR|CACHEDIR|PRIVATE_DIR"
```

е) Использовать для очистки старых БД команду:

```
    smbd -b | grep -E "LOCKDIR|STATEDIR|CACHEDIR|PRIVATE_DIR" |\
awk
    '{print $NF}' | xargs -I'{}' rm -fv '{}'/*.{tdb,ldb}
```

Если эта команда ничего не выдала, то БД от прошлых запусков контроллера домена не было.

ж) Отключить службу systemd-resolved на порту 53, которая может препятствовать работе DNS-сервера, встроенного в Samba или основанного на BIND. Чтобы устранить конфликт, необходимо отключить механизм DNS Stub Listener командой:

```
    mkdir -p /etc/systemd/resolved.conf.d
    echo -e '[Resolve]\nDNSStubListener=no' >\
/etc/systemd/resolved.conf.d/no53port.conf
    sudo systemctl try-restart systemd-resolved
```

После перезапуска systemd-resolved порт 53 освободится и станет доступен для использования контроллером домена.

з) Запустить интерактивную настройку домена:

```
    samba-tool domain provision --use-rfc2307 --interactive
```

Ниже приведен ее типовой вывод. В квадратных скобках ([ ]) указывается значение по умолчанию. Следует нажать клавишу `Enter`, чтобы согласиться с предложенным или ввести иное значение. Если сеть и имя хоста были настроены верно, то нет необходимости изменять значения по умолчанию:

```
    [root@server1 ~]# samba-tool domain provision --use-rfc2307\
--interactive
    Realm [ROSA.LAN]:
    Domain [ROSA]:
    Server Role (dc, member, standalone) [dc]:
    DNS backend (SAMBA_INTERNAL, BIND9_DLZ, BIND9_FLATFILE, NONE)
[SAMBA_INTERNAL]:
    DNS forwarder IP address (write 'none' to disable forwarding)
[192.168.122.1]:
    Administrator password:
```

```
Retype password:
```

На последнем шаге необходимо ввести пароль доменного пользователя Administrator. При вводе пароль не отображается и он должен быть сложным.

Примечания

– При выборе типа DNS backend рекомендуется использовать BIND9_DLZ, если планируется интеграция с внешним DNS-сервером BIND.

– Использование BIND9_FLATFILE не поддерживается и будет исключено из будущих версий Samba.

6.3.3 Перенаправление

DNS Forwarder – это адрес сервера, к которому перенаправляются DNS-запросы, на которые сам контроллер домена не может ответить, например, запрос DNS yandex.ru будет направлен в него.

Для настройки перенаправления нужно провести следующие действия:

а) Выполнить команду:

```
mv -v /var/lib/samba/private/krb5.conf /etc/krb5.conf
```

б) Добавить службу контроллера домена в автозапуск:

```
sudo systemctl enable samba
```

в) Запустить контроллер домена:

```
sudo systemctl restart samba
```

г) Посмотреть лог контроллера домена и убедиться, что он запустился без ошибок:

```
systemctl status samba
```

Статус службы должен быть active (запущена), но в конце лога может быть ошибка, например:

```
dnsupdate_nameupdate_done: Failed DNS update with exit code
29
```

д) Убедиться, что сведения об ошибке не являются критическими, запустив команду:

```
sudo samba_dnsupdate --verbose
```

В подробном логе будет запись про ошибки "WERR_DNS_ERROR_RECORD_ALREADY_EXISTS", которые означают, что добавляемая запись DNS уже существует. Она была создана вместе с доменом.

Запись "Failed update of 29 entries" означает, что такая ситуация возникла 29 раз, что и стало кодом возврата samba_dnsupdate.

е) Проверить работу DNS-сервера на другом компьютере, например, на будущем клиенте домена, выполнив команду:

```
nslookup samba1.rosa.lan 192.168.122.12
```

Для корректного выполнения необходимо подставить в пример реальные IP-адреса.

Пример успешного ответа:

```
$ nslookup samba1.rosa.lan 192.168.122.12
Server: 192.168.122.12
Address: 192.168.122.12#53
Name: samba1.rosa.lan
Address: 192.168.122.12
```

Для настройки клиентов будет полезно знать рабочую группу NT с помощью команды:

```
[root@server1 ~]# cat /etc/samba/smb.conf | grep workgroup =\
SAMBA
```

Следует убедиться, что этот сервер способен выдать DNS требуемого домена, выполнив на будущем клиенте домена (не на контроллере) команду, например:

```
nslookup yandex.ru 192.168.122.1
```

После первоначальной настройки контроллера его можно заменить в файле /etc/samba/smb.conf, перезапустив контроллер после изменения настроек командой:

```
sudo systemctl restart samba
```

После успешной настройки контроллера домена рекомендуется создать хотя бы одного пользователя домена. Это позволит проверить корректность функционирования службы аутентификации и в дальнейшем использовать учётные записи для входа клиентов в домен.

Для создания нового пользователя следует выполнить следующую команду:

```
samba-tool user create <имя_пользователя>
```

Пример:

```
samba-tool user create IvanovVP
```

После ввода команды будет запрошен пароль для новой учётной записи. Пароль должен соответствовать установленной политике безопасности и быть достаточно сложным.

6.3.4 Использование внешнего DNS-сервера BIND

Контроллер домена Samba может использовать встроенный механизм обработки DNS-запросов (SAMBA_INTERNAL) или быть настроен для работы с внешним DNS-сервером BIND. В ряде случаев рекомендуется использовать BIND с поддержкой DLZ-модуля.

Для начала необходимо установить пакет bind:

```
sudo dnf install bind
```

По умолчанию служба BIND настроена на прослушивание только интерфейса loopback (127.0.0.1). Чтобы разрешить обработку запросов на всех сетевых интерфейсах, следует изменить конфигурацию:

```
sudo sed -i '/listen-on\|allow-query/ s|{.\+}|{ any; }|' /etc/named.conf
```

Затем необходимо включить использование DLZ-модуля:

```
echo 'include "/var/lib/samba/bind-dns/named.conf";' >> /etc/named.conf
```

Служба systemd-resolved может занимать порт 53, используемый DNS-сервером BIND. Чтобы устранить конфликт, необходимо отключить DNS Stub Listener:

```
mkdir -p /etc/systemd/resolved.conf.d
echo -e '[Resolve]\nDNSStubListener=no' > /etc/systemd/resolved.conf.d/no53port.conf
sudo systemctl try-restart systemd-resolved
```

6.3.4.1 Запуск службы BIND

Для того чтобы добавить BIND в автозагрузку и запустить службу, используется команда:

```
sudo systemctl enable --now named
```

Для проверки статуса службы:

```
systemctl status named
```

В ходе выполнения команды `samba-tool domain provision` необходимо выбрать тип DNS-сервера, при этом:

– при использовании BIND следует указать BIND9_DLZ в качестве типа DNS backend.

– не рекомендуется использовать BIND9_FLATFILE, так как он не поддерживается и будет удалён в будущих версиях Samba.

6.3.5 Подключение клиента ROSA к домену

Для полноценной работы доменной инфраструктуры необходимо подключить клиентскую машину с ОС РОСА "ХРОМ" к созданному домену. Это позволяет пользователям аутентифицироваться через контроллер домена, а также использовать единые политики безопасности.

Для настройки сети на клиенте следует:

а) убедиться, что клиент находится в одной подсети с контроллером домена и имеет доступ к нему по сети;

б) добавить IP-адрес контроллера домена в список DNS-серверов;

в) добавить имя домена (realm) в список доменов поиска одним из следующих способов:

– через графический интерфейс утилиты NetworkManager необходимо перейти в "Параметры системы → блок "Сеть и связь → Соединения":

1) открыть настройки сетевого подключения;

2) перейти во вкладку "IPv4" (рисунок 96);

3) в поле "DNS-серверы" указать IP-адрес контроллера домена (например, 192.168.122.12);

4) в поле "Домены поиска" указать имя домена (например, samba.loc);

5) сохранить изменения.

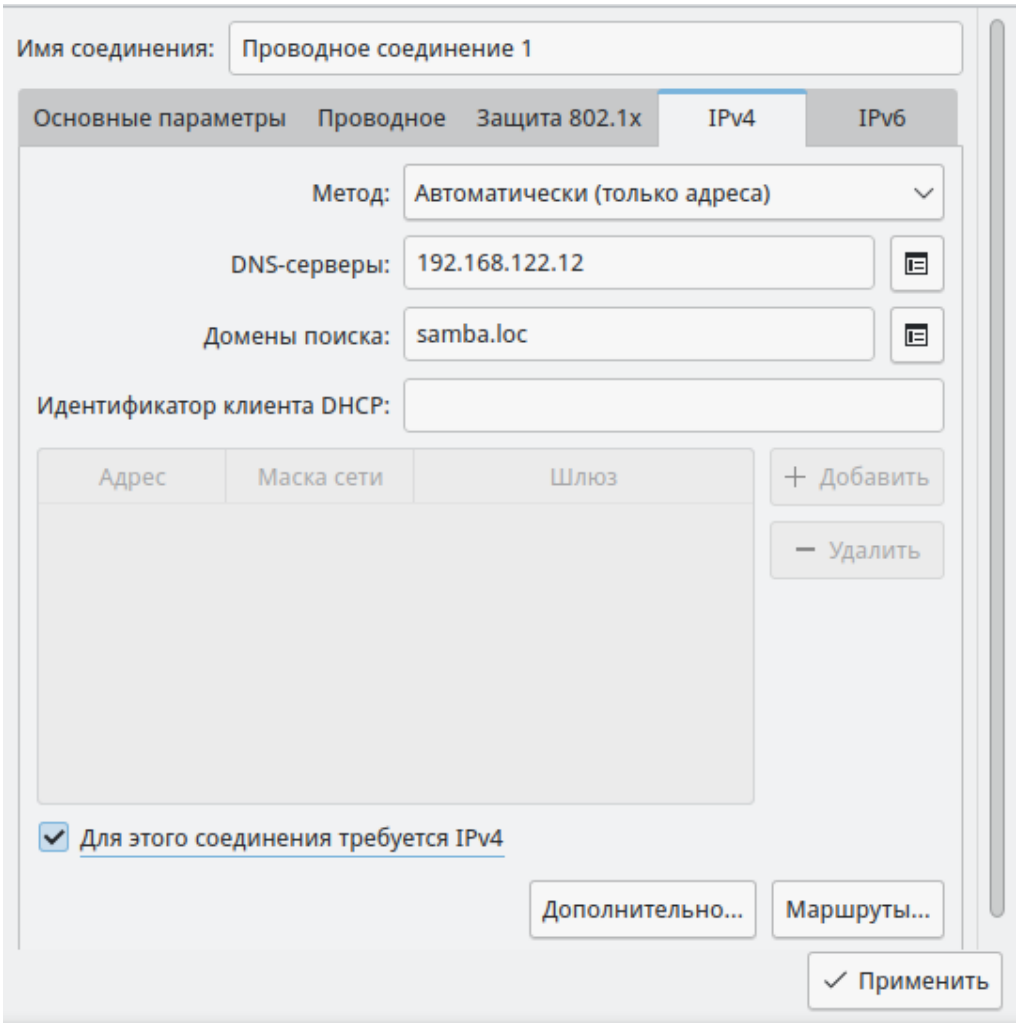


Рисунок 96 – Настройка сети в графическом интерфейсе утилиты NetworkManager

– через утилиту nmtui или вручную:

– в ROSA 2019.05 и новее можно использовать утилиту nmtui;

– при отсутствии NetworkManager можно вручную отредактировать файл /etc/resolv.conf, добавив строки:

```
nameserver 192.168.122.12
search samba.loc
```

– при использовании resolv.conf вручную убедиться, что он не перезаписывается DHCP-клиентом;

г) проверить сетевую доступность контроллера:

```
ping server1.samba.loc
```

д) на клиентской машине установить следующие пакеты:

```
sudo dnf install drakxtools samba-server samba-client samba-winbind nss_ldap libnss-role pam_krb5 lib64sasl2-plug-gssapi urpmi perl-URPM
```

Для подключения к домену с помощью команды `drakauth` необходимо:

а) запустить утилиту `drakauth` командой:

```
drakauth
```

Если переменная окружения $DISPLAY установлена, откроется графический интерфейс, в противном случае будет доступен псевдографический режим;

б) выбрать способ аутентификации – Домен Windows (рисунок 97);

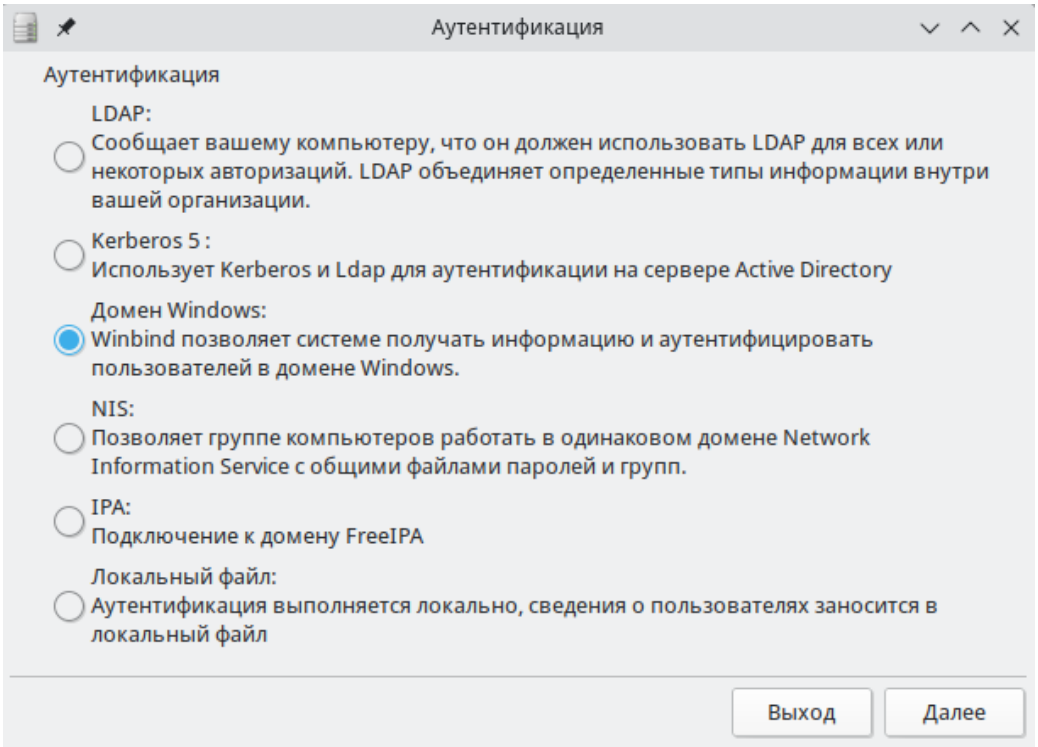


Рисунок 97 – Выбор способа аутентификации в графическом интерфейсе drakauth

в) указать тип домена – Active Directory (рисунок 98);

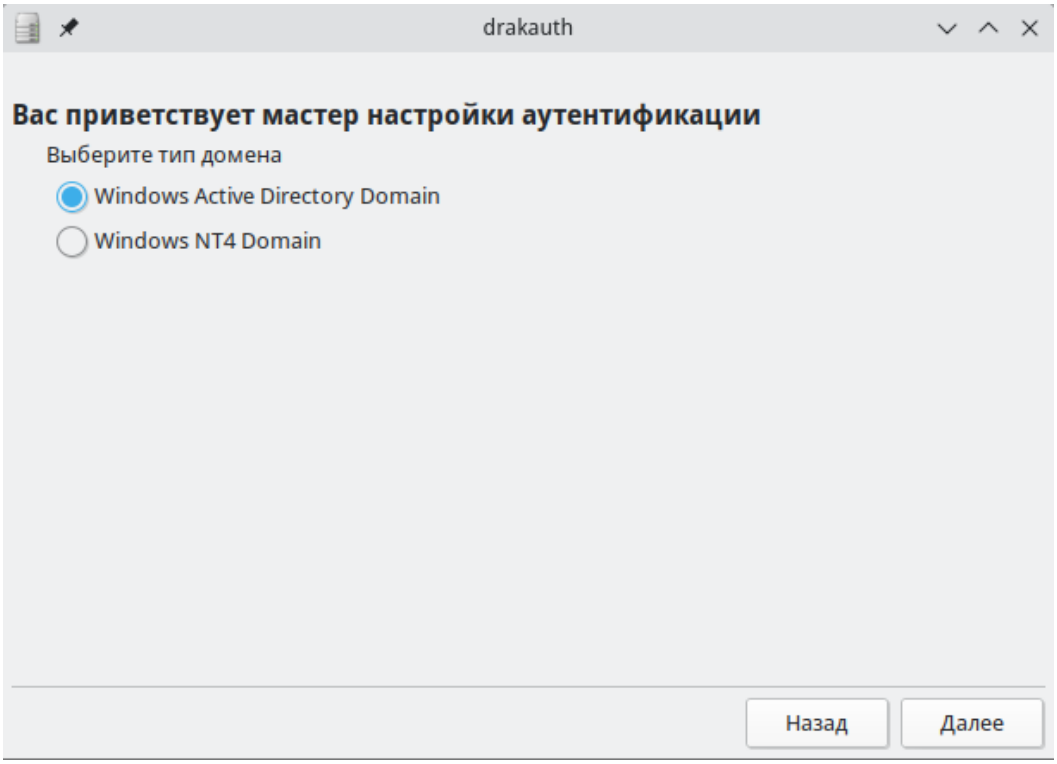


Рисунок 98 – Выбор типа домена в графическом интерфейсе drakauth

г) убедиться, что поля домена и сервера заполнены автоматически (рисунок 99). При необходимости внести корректировки;

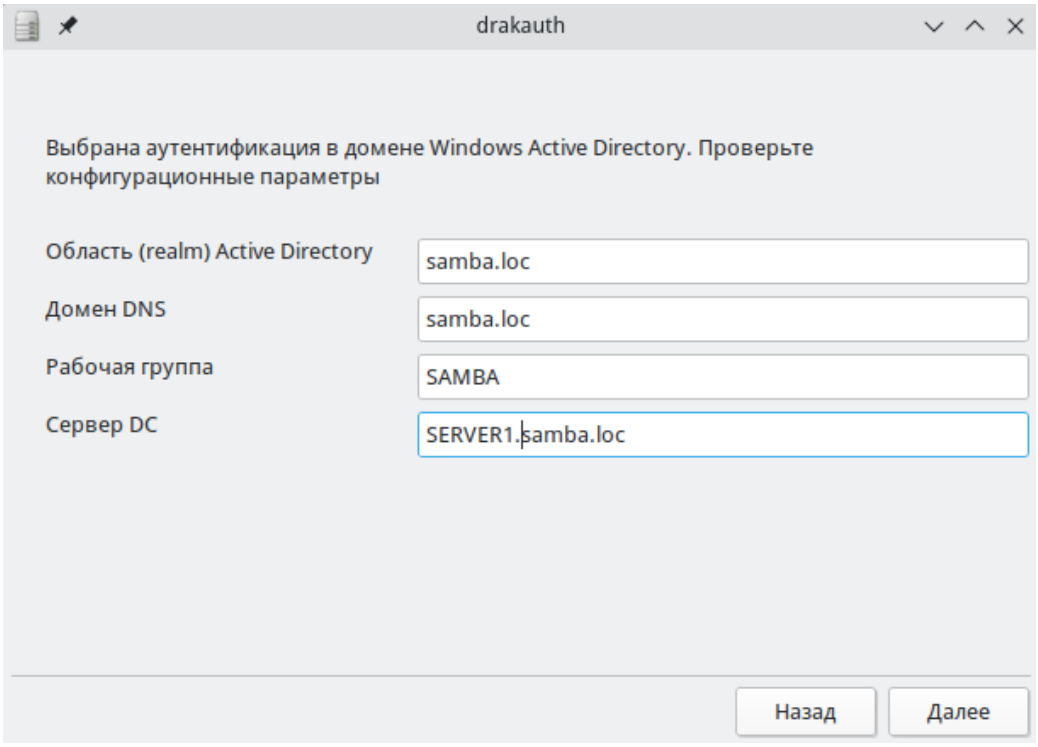


Рисунок 99 – Проверка параметров конфигурации в графическом интерфейсе drakauth

д) указать имя NetBIOS, например, client1; поле "Описание компьютера" заполнять необязательно (рисунок 100);

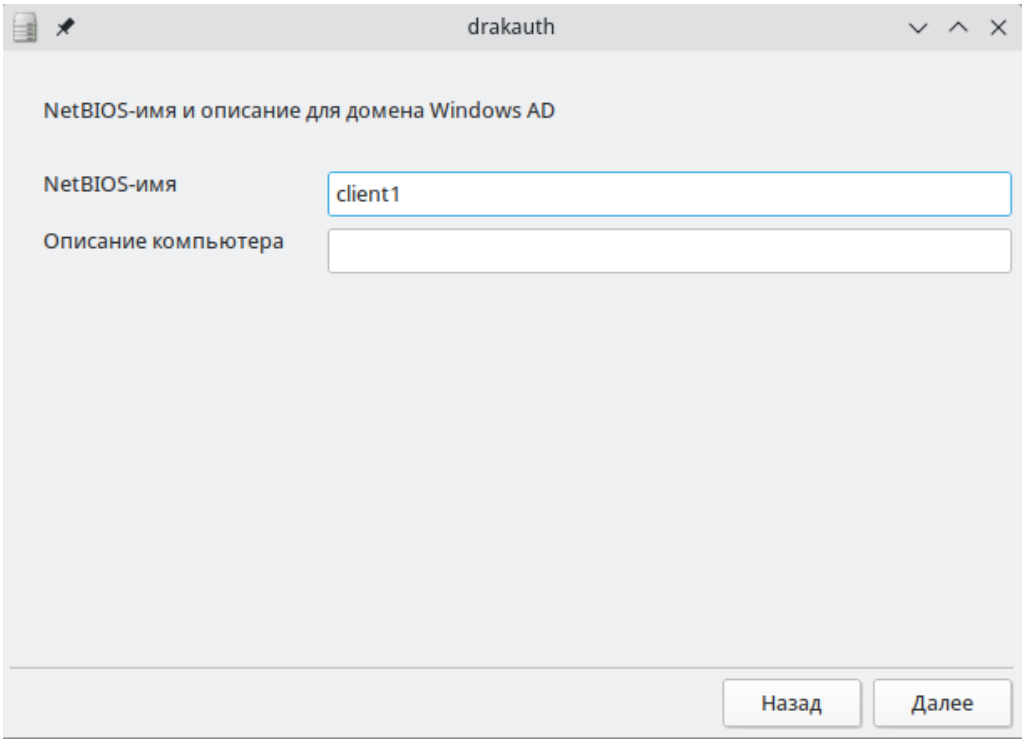


Рисунок 100 – Добавление описания для домена в графическом интерфейсе drakauth

е) ввести логин и пароль администратора домена (рисунок 101);

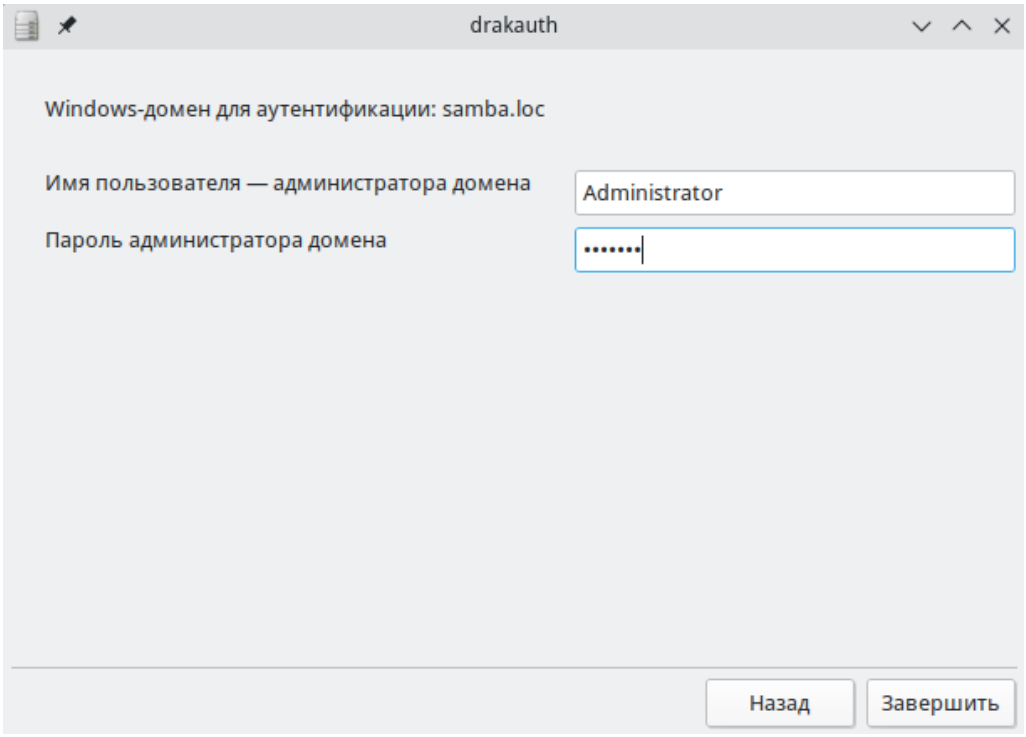


Рисунок 101 – Назначение логина и пароля для администратора домена в графическом интерфейсе drakauth

Если ввод в домен прошел успешно, то будет предложено перезагрузить Систему. В случае возникновения ошибок при подключении домена на экран будет выведено соответствующее окно с сообщением и предложением проверить установленные настройки;

ж) после перезагрузки можно проверить подключение командой:

```
wbinfo -u
```

Команда должна отобразить список доменных пользователей. Также можно проверить вход в Систему под доменной учётной записью.

6.4 Веб-серверы

6.4.1 Apache

Для обеспечения работы веб-сервера используются следующие программные средства:

– Apache HTTP-сервер – веб-сервер с открытым исходным кодом;
– PHP – скриптовый язык программирования, созданный для генерации HTML-страниц на веб-сервере и работы с БД;
– MySQL – свободная система управления базами данных (СУБД);
– PhpMyAdmin – инструмент для визуальной работы с БД MySQL.

6.4.1.1 Установка Apache

Для установки веб-сервера Apache из консоли нужно установить один пакет, который по зависимостям установит еще порядка 40 пакетов, необходимых для нормальной и полноценной работы сервера.

Основные команды для работы с веб-сервером Apache:

– установка пакетов с Apache:

```
sudo dnf install apache-base
```

– запуск сервера Apache:

```
sudo systemctl start httpd
```

– добавление сервера Apache в автозапуск при старте Системы:

```
sudo systemctl enable httpd
```

– остановка сервера:

```
sudo systemctl stop httpd
```

После каких-либо изменений в конфигурационных файлах сервер следует перезапустить:

```
sudo systemctl restart httpd
```

6.4.1.2 Установка PHP

Для работы с PHP нужно установить всего 3 пакета, выполнив команду:

```
sudo dnf install php php-mysql apache-mod_php
```

6.4.1.3 Установка MySQL

Для установки MySQL-сервера нужно выполнить команду:

```
sudo dnf install mysql-server
```

Запуск сервера MySQL:

```
sudo systemctl start mysqld
```

Включение запуска сервера при загрузке компьютера:

```
sudo systemctl enable mysqld
```

Перезапуск сервера:

```
sudo systemctl restart mysqld
```

6.4.1.4 Установка PhpMyAdmin

Перед установкой PhpMyAdmin необходимо проверить корректность предыдущих установок и настроек.

Для безопасности работы с БД MySQL нужно поменять пароль администратора:

```
mysqladmin -u root password новый_пароль
```

Сначала необходимо запустить сервера в правильной последовательности:

```
sudo systemctl start mysqld
sudo systemctl start httpd
```

Теперь можно проверить работоспособность локального сервера, для чего нужно набрать в адресной строке браузера http://localhost/.

Если все прошло удачно, то страница браузера будет отображать сообщение с текстом об успешной работе (рисунок 102).

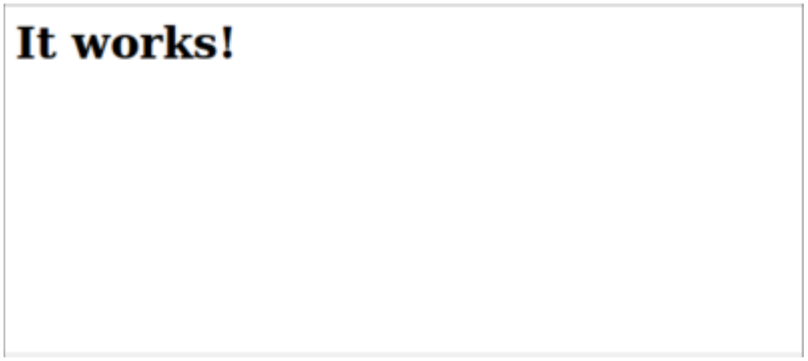


Рисунок 102 – Окно браузера при успешной установке сервера

Затем нужно проверить работу PHP, для этого в папке /var/www/html нужно создать файл info.php с одной строчкой:

```
<?php phpinfo(); ?>
```

Теперь можно проверить работоспособность PHP на локальном сервере, набрав в адресной строке браузера http://localhost/info.php. Окно браузера должно выглядеть так, как показано ниже (рисунок 103).



Рисунок 103 – Проверка работоспособности PHP

Далее следует прокрутить эту страницу ниже и убедиться в работоспособности MySQL. В разделе MySQL должен быть установлен статус Enabled (рисунок 104).

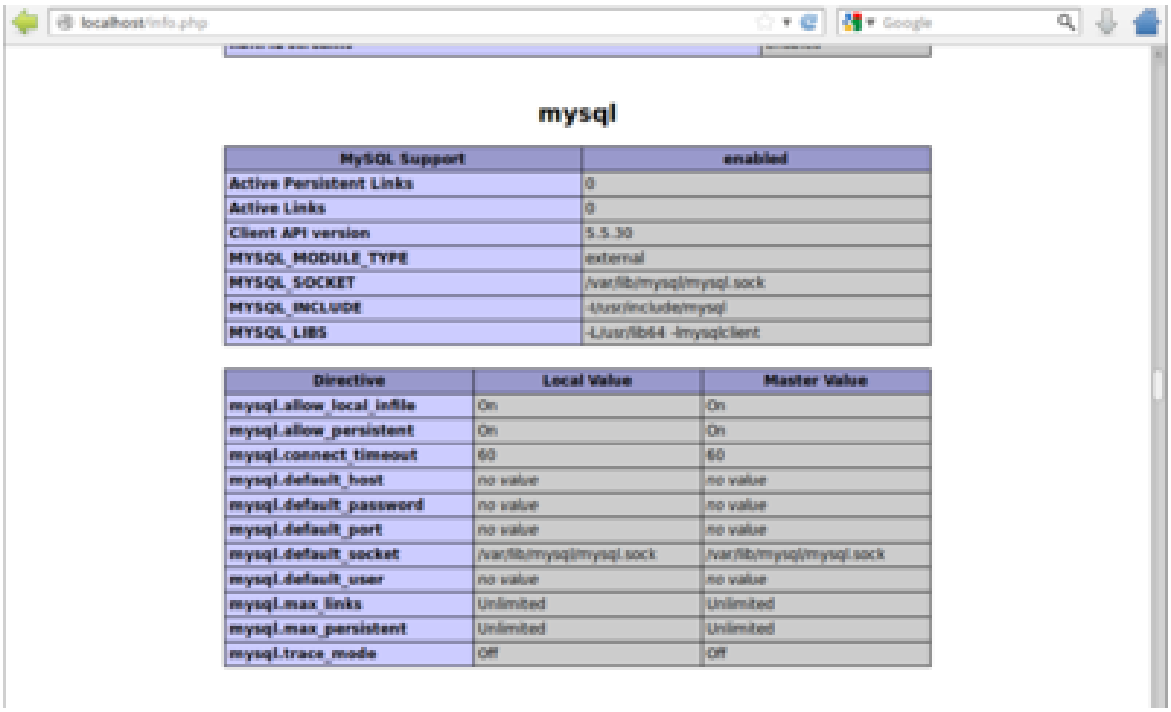


Рисунок 104 – Проверка статуса MySQL

После всех проверок установленных программных средств можно установить PhpMyAdmin командой:

```
sudo dnf install phpmyadmin
```

Затем необходимо заменить содержимое установленного по умолчанию файла /etc/httpd/conf/webapps.d/phpmyadmin.conf на следующее:

```
Alias /phpmyadmin /usr/share/phpmyadmin
<Directory /usr/share/phpmyadmin>
 Options none
 AllowOverride Limit

 Require all granted
</Directory>
```

Чтобы проверить работу PhpMyAdmin, нужно набрать в адресной строке браузера http://localhost/phpmyadmin/.

Если все сделано правильно, то в браузере появится содержание, схожее с представленным ниже (рисунок 105).

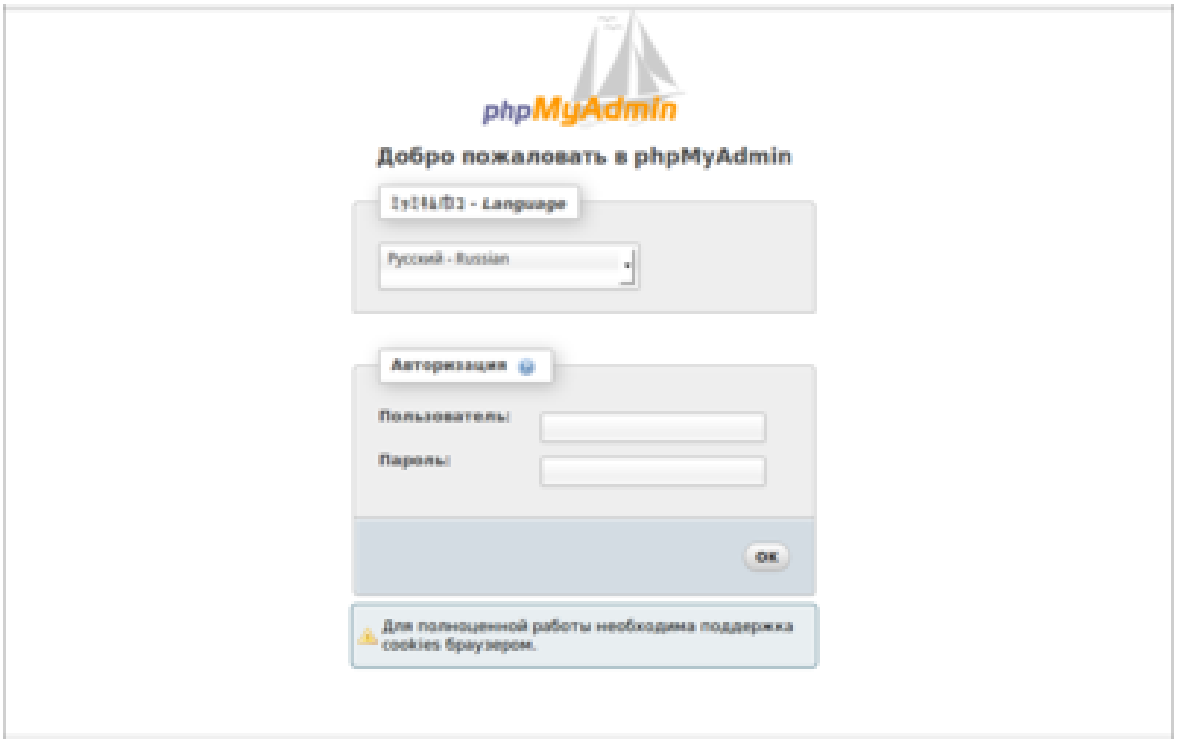


Рисунок 105 – Окно входа в PhpMyAdmin

6.4.2 Веб-сервер Nginx

Nginx — это один из самых популярных веб-серверов, который обеспечивает работу множества высоконагруженных сайтов с большим объёмом трафика. Как правило, Nginx использует ресурсы эффективнее, чем Apache, и может

применяться как полноценный веб-сервер, так и в роли обратного прокси-сервера.

Возможности сервера Nginx:

– обработка статических запросов, индексных файлов; автоматическое формирование списка файлов в директориях; кеширование дескрипторов открытых файлов;

– высокопроизводительное обратное проксирование с возможностью кеширования; поддержка балансировки нагрузки и механизмов отказоустойчивости;

– поддержка FastCGI, uwsgi, SCGI и memcached-серверов с кешированием, с возможностью распределения нагрузки и организации отказоустойчивости;

– модульная архитектура, включая фильтры обработки контента (сжатие gzip, поддержка byte-ranges, chunked-ответов, XSLT-фильтров, SSI-фильтров, преобразование изображений);

– параллельная обработка подзапросов внутри одной страницы через SSI-фильтр с возможностью передачи запросов к сторонним FastCGI- или прокси-серверам;

– поддержка SSL и расширения TLS SNI;

– конфигурация виртуальных серверов по имени хоста или IP-адресу;

– поддержка keep-alive- и pipelined-соединений;

– гибкая система конфигурации;

– возможность изменения настроек и обновления исполняемого файла без прерывания обслуживания клиентов;

– настройка форматов журналов, буферизация и ротация логов;

– отображение специальных страниц ошибок (3xx-5xx);

– модуль rewrite с поддержкой регулярных выражений для перенаправлений;

– ограничение доступа по IP-адресам, HTTP Basic-аутентификация;

– проверка заголовка HTTP referer;

– поддержка методов HTTP: PUT, DELETE, MKCOL, COPY, MOVE;

– потоковая передача мультимедиа (FLV и MP4);

– ограничение скорости ответов и числа одновременных подключений;

– встроенная поддержка Perl-скриптов.

Nginx может использоваться как прокси-сервер для почтовых протоколов с проверкой пользователей через внешний HTTP-сервер аутентификации. Поддерживаются протоколы POP3, IMAP и SMTP с методами авторизации USER/PASS, APOP, AUTH LOGIN/PLAIN/CRAM-MD5. Также реализована поддержка SSL, STARTTLS и STLS для организации защищённых соединений при работе с почтовыми сервисами.

Архитектура Nginx построена на модели одного главного и нескольких рабочих процессов, где рабочие процессы запускаются под непривилегированным пользователем. Для обработки большого количества одновременных соединений используются высокопроизводительные механизмы событий, включая kqueue (FreeBSD 4.1+), epoll (Linux 2.6+), rt signals (Linux 2.2.19+), /dev/poll (Solaris 7+), event ports (Solaris 10), select и poll.

В Nginx поддерживаются дополнительные функции событийных моделей, например: EV_CLEAR, EV_DISABLE, NOTE_LOWAT, EV_EOF и другие; реализована поддержка механизмов передачи файлов (sendfile, sendfile64, sendfilev), асинхронного ввода-вывода (AIO) и режимов DIRECTIO; кроме того, возможно применение accept-фильтров (FreeBSD 4.1+, NetBSD 5.0+) и TCP_DEFER_ACCEPT (Linux 2.4+).

Сервер Nginx демонстрирует низкое потребление памяти (порядка 2,5 МБ на 10000 неактивных keep-alive-соединений) и минимизацию операций копирования данных при передаче запросов, что обеспечивает его высокую производительность и масштабируемость.

Официальный сайт проекта: https://nginx.org/ru/.

6.4.2.1 Установка и структура файлов Nginx

Для установки веб-сервера Nginx необходимо выполнить следующую команду:

```
sudo dnf install nginx
```

После установки требуется запустить сервис Nginx:

```
sudo systemctl start nginx.service
```

Чтобы обеспечить автоматический запуск сервиса Nginx при старте ОС, необходимо выполнить команду:

```
sudo systemctl enable nginx.service
```

Структура файлов и директорий Nginx включает в себя:

– Каталог веб-контента /usr/share/nginx/html. Данная директория содержит веб-контент, доступный по умолчанию сразу после установки Nginx. В неё входит демонстрационная страница, которую выводит установленный по умолчанию веб-сервер. При необходимости размещения собственного контента или веб-приложений следует отредактировать конфигурацию и указать другую директорию.

– Каталоги конфигураций сервера:

– /etc/nginx/ – основной каталог, содержащий все файлы конфигурации Nginx;

– /etc/nginx/nginx.conf – главный конфигурационный файл, определяющий глобальные параметры работы веб-сервера. Для изменения параметров, применяемых ко всем виртуальным хостам, необходимо вносить правки именно в этот файл;

– /etc/nginx/conf.d/ – каталог, содержащий конфигурационные файлы для отдельных виртуальных хостов. Рекомендуется создавать для каждого сайта отдельный конфигурационный файл с расширением .conf, совпадающий по названию с доменным именем, например, example.ru.conf;

– Каталоги журналов работы:

– /var/log/nginx/access.log – все запросы к веб-серверу фиксируются в данном файле, если в конфигурации Nginx не указано иное;

– /var/log/nginx/error.log – все ошибки и сообщения о сбоях работы веб-сервера регистрируются в этом журнале.

6.4.2.2 Настройка Nginx

Основным конфигурационным файлом Nginx является /etc/nginx/nginx.conf. При необходимости изменения настроек следует открыть данный файл и внести требуемые параметры.

После запуска сервиса Nginx рекомендуется проверить его работоспособность, открыв веб-браузер и введя адрес:

```
http://<IP-адрес>
```

где <IP-адрес> — это адрес сервера, на котором запущен Nginx.

Ниже приведён пример базовой конфигурации /etc/nginx/nginx.conf по умолчанию:

```
#user  nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}
```

```
    http {
        include       mime.types;
        default_type  application/octet-stream;

        #log_format     main     '$remote_addr   -   $remote_user\
[$time_local] "$request" '
        #              '$status $body_bytes_sent "$http_referer"\
'
        #                                    '"$http_user_agent"\
"$http_x_forwarded_for"';

        #access_log  logs/access.log  main;

        sendfile        on;
        #tcp_nopush     on;

        #keepalive_timeout  0;
        keepalive_timeout  65;

        #gzip  on;

        server {
            listen       80;
            server_name  localhost;

            #charset koi8-r;

            #access_log  logs/host.access.log  main;

            location / {
                root   html;
                index  index.html index.htm;
            }

            #error_page  404              /404.html;
```

```
            # redirect server error pages to the static page\
/50x.html
            #
            error_page   500 502 503 504  /50x.html;
            location = /50x.html {
                root   html;
            }

            # proxy the PHP scripts to Apache listening on\
127.0.0.1:80
            #
            #location ~ \.php$ {
            #    proxy_pass   http://127.0.0.1;
            #}

            # pass the PHP scripts to FastCGI server listening
on\ 127.0.0.1:9000
            #
            #location ~ \.php$ {
            #    root           html;
            #    fastcgi_pass   127.0.0.1:9000;
            #    fastcgi_index  index.php;
            #                fastcgi_param       SCRIPT_FILENAME\
/scripts$fastcgi_script_name;
            #    include        fastcgi_params;
            #}

            # deny access to .htaccess files, if Apache's\
document root
            # concurs with nginx's one
            #
            #location ~ /\.ht {
            #    deny  all;
            #}
        }
```

```
        # another virtual host using mix of IP-, name-, and port-
based configuration
        #
        #server {
        #    listen       8000;
        #    listen       somename:8080;
        #    server_name  somename  alias  another.alias;

        #    location / {
        #        root   html;
        #        index  index.html index.htm;
        #    }
        #}


        # HTTPS server
        #
        #server {
        #    listen       443 ssl;
        #    server_name  localhost;

        #    ssl_certificate      cert.pem;
        #    ssl_certificate_key  cert.key;

        #    ssl_session_cache    shared:SSL:1m;
        #    ssl_session_timeout  5m;

        #    ssl_ciphers  HIGH:!aNULL:!MD5;
        #    ssl_prefer_server_ciphers  on;

        #    location / {
        #        root   html;
        #        index  index.html index.htm;
        #    }
        #}

    }
```

где:

– user – определяет пользователя, от имени которого запускается процесс Nginx. В дистрибутиве ОС данный параметр закомментирован, поскольку сервер запускается под пользователем nginx по умолчанию;

– listen – задаёт порт, на котором принимает соединения сервер (например, 80 для HTTP);

– server_name – определяет имя сервера; по умолчанию используется localhost;

– location / – определяет обработку запросов к корневому адресу сайта;

– root – путь к каталогу, из которого осуществляется выдача файлов сайта;

– index – список файлов, которые будут открываться по умолчанию при отсутствии указания конкретного имени файла в запросе.

Настройка виртуальных хостов

Для организации обслуживания нескольких сайтов (виртуальных хостов) рекомендуется выполнять следующие шаги (рассмотрен простой пример для двух сайтов):

а) подготовить структуру каталогов. В каталоге /var/www рекомендуется создать отдельные подкаталоги для каждого сайта, а внутри — директории html для хранения файлов сайта. Например:

```
mkdir -p /var/www/rosaserver1/html
mkdir -p /var/www/rosaserver2/html
```

б) установить владельца каталогов для текущего пользователя с использованием переменной окружения $USER:

```
chown -R $USER:$USER /var/www/rosaserver1/html
chown -R $USER:$USER /var/www/rosaserver2/html
```

в) для редактирования содержимого первого сайта открыть файл командой (для проверки корректности настройки рекомендуется создать простые страницы):

```
nano /var/www/rosaserver1/html/index.html
```

г) добавить в файл следующий код:

```
<html>
<head>
<title>Welcome to RosaServer1!</title>
</head>
<body>
Success! The RosaServer1 block is working!
</body>
</html>
```

д) для редактирования содержимого второго сайта открыть файл командой:

```
nano /var/www/rosaserver2/html/index.html
```

е) добавить в файл следующий код:

```
<html>
<head>
<title>Welcome to RosaServer2!</title>
</head>
<body>
Success! The RosaServer2 block is working!
</body>
</html>
```

ж) файлы необходимо сохранить и закрыть.

Организация структуры виртуальных хостов

Для хранения конфигураций виртуальных хостов требуется создать директории:

```
sudo mkdir /etc/nginx/sites-available
sudo mkdir /etc/nginx/sites-enabled
```

После этого необходимо указать в конфигурационном файле Nginx, что активные блоки server находятся в каталоге sites-enabled. Для этого следует открыть файл /etc/nginx/nginx.conf и в конце блока http {} добавить:

```
include /etc/nginx/sites-enabled/*.conf;
```

Создание файлов конфигурации виртуальных хостов

В качестве шаблона может быть использован файл /etc/nginx/conf.d/virtual.conf. Рекомендуется скопировать его в каталог sites-available для каждого сайта:

```
cp /etc/nginx/conf.d/virtual.conf /etc/nginx/sites-available/rosaserver1.conf
cp /etc/nginx/conf.d/virtual.conf /etc/nginx/sites-available/rosaserver2.conf
```

После копирования необходимо открыть соответствующие файлы и отредактировать их. Например, для первого сайта:

```
sudo nano /etc/nginx/sites-available/rosaserver1.conf
```

Пример содержимого файла:

```
server {
```

```
        listen       80;
        server_name  rosaserver1;

        location / {
            root   /var/www/rosaserver1/html;
            index  index.html index.htm;
        }
    }
```

Для второго сайта нужно произвести аналогичные действия, изменяя параметры server_name и root:

```
    server {
        listen       80;
        server_name  rosaserver2;

        location / {
            root   /var/www/rosaserver2/html;
            index  index.html index.htm;
        }
    }
```

Файлы необходимо сохранить и закрыть.

Включение виртуальных хостов

Для активации виртуальных хостов требуется создать символические ссылки из каталога sites-available в sites-enabled:

```
    ln      -s      /etc/nginx/sites-available/rosaserver1.conf
/etc/nginx/sites-enabled/rosaserver1.conf
    ln      -s      /etc/nginx/sites-available/rosaserver2.conf
/etc/nginx/sites-enabled/rosaserver2.conf
```

Необходимо выполнить перезапуск сервиса Nginx, чтобы применить новые настройки:

```
    sudo systemctl restart nginx.service
```

Чтобы проверить работу виртуальных хостов, рекомендуется открыть веб-браузер и проверить доступность созданных сайтов по адресам:

```
    http://rosaserver1/
    http://rosaserver2/
```

При корректной настройке должны отобразиться созданные ранее демонстрационные страницы.

6.4.2.3 Настройка HTTPS/TLS в Nginx

Поддержка защищённых соединений с использованием протоколов SSL/TLS является одной из важнейших функций веб-сервера Nginx. Реализация HTTPS позволяет обеспечить конфиденциальность и целостность данных при обмене между пользователем и сервером.

Nginx предоставляет гибкие возможности для настройки TLS, включая выбор версий протокола, шифров, поддержку OCSP Stapling и других расширений безопасности.

Для тестовых целей или при отсутствии центра сертификации может использоваться самоподписанный сертификат. Для его создания необходимо выполнить следующую команду:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/ssl/private/server.key \
-out /etc/ssl/certs/server.crt
```

В результате будет создан сертификат server.crt и соответствующий ему закрытый ключ server.key. Необходимо обеспечить надёжные права доступа к закрытому ключу, например:

```
chmod 600 /etc/ssl/private/server.key
```

Пример базовой конфигурации TLS в Nginx:

```
server {
    listen              443 ssl;
    server_name         example.local;

    ssl_certificate     /etc/ssl/certs/server.crt;
    ssl_certificate_key /etc/ssl/private/server.key;

    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    ssl_prefer_server_ciphers on;
    ssl_session_cache   shared:SSL:10m;
    ssl_session_timeout 10m;

    location / {
        root   /var/www/html;
        index  index.html index.htm;
    }
```

```
}
```

где:

– ssl_protocols – перечень протоколов, разрешённых для использования. В целях безопасности рекомендуется ограничиться TLSv1.2 и TLSv1.3;

– ssl_ciphers – определяет допустимые наборы шифров. Рекомендуется использовать HIGH с исключением слабых алгоритмов, например, HIGH:!aNULL:!MD5;

– ssl_prefer_server_ciphers – включает приоритет серверных шифров при согласовании алгоритма между клиентом и сервером;

– ssl_session_cache – определяет параметры кеширования сессий TLS, что уменьшает время повторных подключений;

– ssl_session_timeout – задаёт время жизни кешированной сессии.

После изменения конфигурации необходимо выполнить проверку синтаксиса командой:

```
sudo nginx -t
```

Далее нужно перезапустить сервис для применения изменений:

```
sudo systemctl reload nginx
```

6.4.2.4 Настройка обратного проксирования в Nginx

Веб-сервер Nginx нередко используется не только для раздачи статического контента, но и в качестве высокопроизводительного обратного прокси-сервера (reverse proxy), который перенаправляет запросы на другие приложения, например: Apache, Tomcat, Node.js или другие веб-сервисы. Такой подход позволяет централизовать обработку SSL, уменьшить нагрузку на сервер приложений, реализовать балансировку нагрузки и повысить отказоустойчивость.

Пример, при котором Nginx принимает запросы на порт 80 и перенаправляет их на внутренний сервер Apache, работающий на порту 8080:

```
server {
    listen       80;
    server_name  example.local;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
            proxy_set_header                    X-Forwarded-For\
$proxy_add_x_forwarded_for;
        }
    }
```

В данной конфигурации:

– все запросы отправляются на 127.0.0.1:8080 (Apache);

– передаются заголовки с исходным IP-адресом пользователя и именем хоста для правильной обработки логики приложения;

– сохраняется возможность ведения журналов с реальными IP пользователей.

Директива proxy_pass указывает, куда именно будет перенаправляться запрос. Поддерживается схема http://, https:// или даже unix:/path/to/socket.

Например, следующая команда означает, что все запросы из данного блока location перенаправляются на указанный адрес:

```
    proxy_pass http://backend-app:8000;
```

При наличии нескольких серверов приложений можно использовать блок upstream для балансировки запросов, например:

```
    upstream backend {
        server 192.168.1.10;
        server 192.168.1.11;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
        }
    }
```

В этом примере Nginx будет распределять запросы по серверам 192.168.1.10 и 192.168.1.11 по умолчанию методом round-robin.

Для более сложных сценариев доступны опции sticky-сессий, настройки fail_timeout и другие механизмы управления доступностью бэкендов.

Рекомендуется явно задать тайм-ауты для взаимодействия с проксируемыми серверами, чтобы избежать зависания соединений:

```
    proxy_connect_timeout       5s;
    proxy_send_timeout         30s;
    proxy_read_timeout         30s;
    send_timeout              30s;
```

Эти параметры позволяют ограничить время установления соединения, время отправки и получения данных от сервера приложений, что существенно повышает стабильность работы при сетевых сбоях или ошибках на стороне бэкенда.

После внесения изменений необходимо проверить корректность конфигурации с помощью команды:

```
    sudo nginx -t
```

Далее нужно перезапустить веб-сервер:

```
    sudo systemctl reload nginx
```

6.4.2.5 Механизмы защиты Nginx

Для повышения уровня безопасности веб-сервера Nginx рекомендуется применять ряд механизмов, которые помогут защитить сервис от перегрузок, злоупотреблений и атак типа DoS. К таким механизмам относятся:

– **Ограничение числа одновременных подключений и скорости**:

– Настройка ограничения числа подключений:

```
    limit_conn_zone $binary_remote_addr zone=addr:10m;
    limit_conn addr 20;
```

– Настройка ограничения скорости запросов:

```
    limit_req_zone    $binary_remote_addr    zone=req_zone:10m\
rate=5r/s;
    limit_req zone=req_zone burst=10 nodelay;
```

Такая конфигурация позволит производить не более 20 одновременных соединений с одного IP-адреса, а также не более 5 запросов в секунду.

– **Защита от слишком больших запросов**, которые могут вызвать отказ сервиса. Рекомендуется задать максимальный размер тела запроса:

```
    client_max_body_size 2m;
```

Значение может быть указано в Мб или Кб в зависимости от предполагаемой нагрузки.

– **Интеграция с внешними инструментами фильтрации**, такими как fail2ban.

Fail2ban — это утилита, позволяющая отслеживать в журналах сервера подозрительные попытки входа или злоупотребления и автоматически блокировать IP-адреса злоумышленников через брандмауэр.

Для защиты Nginx обычно используется фильтр, анализирующий ошибки в файле журнала. Для установки и настройки фильтра необходимо выполнить следующие действия:

а) установить пакет fail2ban;

б) создать фильтр, например, /etc/fail2ban/filter.d/nginx-http-auth.conf, с выражениями для поиска атак;

в) настроить jail (правило) в /etc/fail2ban/jail.local, например:

```
[nginx-http-auth]
enabled  = true
filter   = nginx-http-auth
port     = http,https
logpath  = /var/log/nginx/error.log
maxretry = 5
findtime = 600
bantime  = 3600
```

После этого fail2ban будет блокировать IP-адреса, с которых происходят слишком частые ошибки аутентификации или другие подозрительные действия.

В комплексе с настройками fail2ban стоит использовать внутренние директивы Nginx, которые позволяют реализовать базовую защиту от DoS-атак:

– limit_conn – ограничивает число одновременных соединений на одного клиента;

– limit_req – ограничивает частоту запросов.

Они могут комбинироваться, создавая эффективный барьер против перегрузок сервиса и злоупотреблений.

После внесения изменений рекомендуется проверить корректность конфигурации:

```
sudo nginx -t
```

Затем нужно применить её:

```
sudo systemctl reload nginx
```

6.4.2.6 Логирование и анализ в Nginx

Настроенные журналы логирования позволяют выявлять проблемы, отслеживать попытки несанкционированного доступа и анализировать нагрузку.

Рекомендуется выполнить комплексную настройку логирования по следующим этапам:

а) Настройка формата access.log. По умолчанию журнал запросов access.log ведётся с базовым форматом, который может быть расширен под конкретные требования. Для этого в конфигурации определяется директива `log_format`:

```
    log_format main '$remote_addr - $remote_user [$time_local]\
"$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent"
"$http_x_forwarded_for"';
```

Эта конструкция позволит фиксировать IP-адрес клиента, дату, строку запроса, статус ответа, реферер и заголовки агента.

б) Использование пользовательских форматов журналирования (custom log format). Если требуется регистрировать дополнительные данные, например, время ответа или значения cookie, можно добавить их в шаблон `log_format`:

```
    log_format custom '$remote_addr [$time_local] "$request" '
                      'status=$status bytes=$body_bytes_sent '
                      'rt=$request_time cookie="$http_cookie"';
```

После чего необходимо указать данный формат в директиве `access_log`:

```
    access_log /var/log/nginx/access.log custom;
```

в) Ротация логов с использованием logrotate. Для автоматического ограничения размера журналов и предотвращения переполнения файловой системы используется система logrotate. В дистрибутивах ОС конфигурация логирования Nginx обычно хранится в файле /etc/logrotate.d/nginx. Пример типичной настройки:

```
    /var/log/nginx/*.log {
        daily
        missingok
        rotate 7
        compress
        delaycompress
```

```
        notifempty
        create 0640 nginx adm
        sharedscripts
        postrotate
           sudo systemctl reload nginx.service > /dev/null 2>&1
|| true
        endscript
    }
```

Такая конфигурация обеспечивает ежедневную ротацию, хранение семи архивных копий, сжатие старых журналов и автоматическую перезагрузку службы после ротации.

В примере типичной настройки:

– daily – выполняется ротация журналов ежедневно;
– rotate 7 – хранить 7 предыдущих архивных копий;
– compress – при ротации старые файлы сжимаются с помощью gzip;
– delaycompress – откладывает сжатие на один цикл ротации, чтобы не сжимать самый новый архив сразу;
– postrotate ... endscript – блок команд, который выполняется после ротации; в данном случае перезапускает конфигурацию сервиса Nginx для корректного использования нового файла журнала.

Интеграция с централизованной системой журналирования. При необходимости можно перенаправить журналы Nginx в syslog или системный журнал journald. Для этого в конфигурации сервера указывается:

```
    access_log
syslog:server=unix:/dev/log,facility=local7,tag=nginx_access;
    error_log
syslog:server=unix:/dev/log,facility=local7,tag=nginx_error warn;
```

6.4.2.7 Резервное копирование и восстановление Nginx

Для обеспечения стабильной и безопасной работы веб-сервера Nginx необходимо предусмотреть процедуры регулярного резервного копирования его конфигурационных файлов, а также описать порядок их восстановления.

Рекомендуется выполнять следующие действия:

а) Определить критически важные файлы и каталоги для резервного копирования. В резервные копии должны обязательно включаться:

– основной файл конфигурации /etc/nginx/nginx.conf;

– все файлы виртуальных хостов, расположенные в каталогах /etc/nginx/conf.d/, /etc/nginx/sites-available/ и /etc/nginx/sites-enabled/;

– пользовательские SSL-сертификаты и ключи в /etc/ssl/certs/ и /etc/ssl/private/;

– при необходимости скрипты и шаблоны мониторинга или логирования, интегрированные с Nginx.

б) Создавать сжатую копию конфигурации и сертификатов не реже одного раза в неделю с сохранением нескольких предыдущих версий архива, используя команду:

```
sudo tar czf /backup/nginx-config-$(date +%Y%m%d).tar.gz /etc/nginx /etc/ssl
```

Такой архив может храниться на отдельном сервере резервного копирования, в том числе с применением средств шифрования при передаче или хранении.

в) Для восстановления после сбоя необходимо выполнить:

– распаковку резервного архива в стандартные каталоги:

```
sudo tar xzf nginx-config-20250620.tar.gz -C /
```

– проверку корректности прав доступа и владельцев файлов;

– проверку конфигурации веб-сервера командой:

```
sudo nginx -t
```

– перезагрузку службы для применения восстановленной конфигурации:

```
sudo systemctl reload nginx
```

При использовании системы управления версиями (например, Git) возможно организовать дополнительный контроль изменений и отслеживать историю правок в конфигурационных файлах.

6.4.2.8 Обновление и управление версиями Nginx

Для контроля актуальности установленной версии рекомендуется:

– проверять текущую версию командой `nginx -v`;

– при необходимости уточнять доступные обновления в репозитории с помощью следующей команды:

```
dnf info nginx
```

Процесс обновления Nginx выполняется стандартным менеджером пакетов dnf:

```
sudo dnf update nginx
```

После обновления или изменения конфигурационных файлов необходимо выполнить безопасную перезагрузку службы с сохранением активных соединений:

```
sudo nginx -s reload
```

Этот вариант позволяет без прерывания обслуживания клиентов применить новую конфигурацию или загрузить обновлённый бинарный файл.

6.4.3 Веб-сервер Angie

Веб-сервер Angie представляет собой современный, эффективный и масштабируемый программный продукт, реализованный как ответвление Nginx. Angie создан бывшими разработчиками исходного проекта с целью развития решения в новом направлении. При этом Angie сохраняет совместимость с конфигурацией и модулями Nginx, что позволяет использовать его в качестве замены без необходимости доработки существующих настроек.

Angie включает все возможности версии Nginx 1.25.3 и расширяет их рядом дополнительных функций, среди которых:

– Поддержка HTTP/3 как для клиентских соединений, так и для соединений с проксируемыми серверами с возможностью раздельного использования различных протоколов (HTTP/1.x, HTTP/2, HTTP/3) на разных направлениях.

– Упрощение конфигурации за счёт возможности в директиве `location` задавать несколько строк сопоставления и объединять блоки с одинаковыми настройками.

– Получение базовой информации о веб-сервере, его конфигурации, а также статистики по клиентским соединениям, проксируемым серверам, зонам разделяемой памяти и другим параметрам через REST-подобный API-интерфейс в формате JSON.

– Возможность экспорта метрик и статистики в формате Prometheus с гибкими шаблонами.

– Интеграция с визуальной консолью мониторинга Console Light, доступной через веб-браузер (онлайн-пример консоли доступен по адресу: https://console.angie.software/).

– Автоматическое обновление списка проксируемых серверов по доменным именам, включая возможность получения данных из DNS-записей SRV.

– Режим привязки сессий, позволяющий направлять все запросы в рамках одной сессии на один и тот же проксируемый сервер.

– Реализация плавного ввода проксируемого сервера после сбоя с использованием опции slow_start директивы `server`.

– Ограничение скорости отдачи MP4-файлов с учётом их битрейта, что снижает нагрузку на сеть.

– Поддержка директивы `mqtt_preread` в модуле stream, расширяющая возможности балансировки и аутентификации по протоколу MQTT.

– Наличие готовых бинарных пакетов для интеграции со многими сторонними модулями.

Официальный сайт проекта: https://angie.software/

6.4.3.1 Установка Angie

Для установки веб-сервера Angie необходимо выполнить следующие действия:

а) установить пакет Angie из репозитория:

```
sudo dnf install angie
```

б) запустить сервис веб-сервера:

```
sudo systemctl start angie.service
```

в) включить автоматический запуск при загрузке:

```
sudo systemctl enable angie.service
```

6.4.3.2 Настройка и управление веб-сервером Angie

По умолчанию структура каталогов Angie совпадает с Nginx:

– /usr/share/angie/html – директория веб-контента;
– /etc/angie/ – основной каталог конфигурации;
– /etc/angie/angie.conf – главный конфигурационный файл;
– /etc/angie/conf.d/ – каталог для виртуальных хостов;
– /var/log/angie/access.log – журнал запросов;
– /var/log/angie/error.log – журнал ошибок.

Настройка веб-сервера Angie осуществляется по тем же принципам, что и настройка Nginx, благодаря полной совместимости конфигурационного синтаксиса. При этом могут использоваться как стандартные подходы конфигурации, так и расширенные возможности, добавленные в Angie.

После установки рекомендуется проверить работу веб-сервера. Для этого следует открыть браузер и перейти по адресу сервера (например, http://<IP-адрес>). При корректном запуске будет отображена стандартная страница Angie.

Основным конфигурационным файлом Angie является /etc/angie/angie.conf. В этом файле задаются глобальные параметры работы сервера.

Рекомендуется придерживаться следующего подхода:

– все изменения в глобальных настройках вносить в angie.conf;

– файлы конфигурации отдельных виртуальных хостов хранить в каталоге /etc/angie/conf.d/;

– при большом числе сайтов структурировать конфигурацию с помощью отдельных директорий sites-available и sites-enabled аналогично практике Nginx.

Для активации виртуальных хостов может быть добавлена директива `include /etc/angie/sites-enabled/*.conf`; в блок http {} основного конфигурационного файла.

Ниже приведён пример минимальной конфигурации виртуального хоста для Angie:

```
server {
    listen       80;
    server_name  example.local;

    location / {
        root   /var/www/example.local/html;
        index  index.html index.htm;
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/angie/html;
    }
}
```

В приведённой конфигурации задаётся имя сервера, корневая директория сайта и базовая обработка ошибок.

После изменения конфигурационных файлов необходимо произвести проверку корректности синтаксиса:

```
sudo angie -t
```

В случае отсутствия ошибок конфигурацию рекомендуется загрузить без прерывания активных соединений:

```
sudo systemctl reload angie.service
```

6.4.3.3 Дополнительные функции Angie

Angie расширяет традиционные функции Nginx следующими механизмами, которые могут быть востребованы при настройке:

– slow_start — позволяет плавно вводить проксируемый сервер в работу после сбоя. Настраивается в блоке upstream:

```
upstream backend {
    server 192.168.1.100;
    server 192.168.1.101;
    slow_start 30s;
}
```

В данном примере повторное включение сервера будет происходить в течение 30 секунд с постепенным увеличением доли запросов.

– `mqtt_preread` — директива для протокола MQTT, применяемая в модуле stream:

```
stream {
    mqtt_preread on;
    server {
        listen 1883;
        proxy_pass mqtt_backend;
    }
}
```

`mqtt_preread` используется для расширенной аутентификации и балансировки MQTT-соединений.

– HTTP/3 — поддержка нового протокола включается в секции listen:

```
server {
    listen 443 quic reuseport;
    http3 on;
    ssl_certificate /etc/ssl/certs/server.crt;
    ssl_certificate_key /etc/ssl/private/server.key;
    ...
}
```

Здесь параметр quic включает использование UDP и HTTP/3, при этом по-прежнему возможен fallback на HTTP/1.x или HTTP/2.

6.4.3.4 Мониторинг и API

Angie предоставляет REST-подобный API для получения данных о состоянии сервера, текущей конфигурации, статистике по зонам разделяемой памяти, соединениям и проксируемым серверам.

В ответах API используется формат JSON, что позволяет легко интегрировать Angie с системами мониторинга. Также поддерживается экспорт метрик Prometheus с гибкой настройкой шаблонов.

Для удобного визуального контроля может использоваться консоль Console Light. Её работа не требует дополнительной установки – необходимо лишь активировать соответствующий блок в конфигурации и открыть веб-интерфейс по адресу сервера.

6.4.3.5 Настройка HTTPS/TLS в Angie

Angie поддерживает все основные механизмы TLS/SSL, применяемые в современных веб-серверах. При этом полностью сохраняется совместимость с синтаксисом Nginx.

Для включения защищённых соединений необходимо выполнить следующие шаги:

– сгенерировать ключ и сертификат (например, с помощью OpenSSL);
– разместить их в защищённой директории, доступной только пользователю, от имени которого работает Angie;
– описать соответствующую секцию server с параметрами listen и ssl_certificate.

Пример минимальной конфигурации HTTPS:

```
server {
    listen              443 ssl;
    server_name         example.local;

    ssl_certificate     /etc/ssl/certs/server.crt;
    ssl_certificate_key /etc/ssl/private/server.key;

    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    location / {
        root /var/www/example.local/html;
        index index.html index.htm;
```

```
        }
    }
```

В данном примере задаются:

– сертификат сервера (ssl_certificate);
– закрытый ключ (ssl_certificate_key);
– допустимые протоколы TLS (рекомендуется ограничить tlsv1.2 и tlsv1.3);
– набор безопасных шифров (ssl_ciphers).

Рекомендуется придерживаться следующих принципов для безопасной настройки веб-сервера:

– сертификаты и ключи следует хранить в директории с ограниченными правами доступа;
– не использовать SSL-протоколы ниже tlsv1.2;
– при работе с публичными центрами сертификации (ca) важно настроить ocsp stapling для повышения надёжности;
– желательно активировать параметр ssl_prefer_server_ciphers on, чтобы сервер контролировал выбор шифра.

После внесения всех изменений в конфигурационные файлы необходимо выполнить проверку синтаксиса:

```
sudo angie -t
```

После проверки необходимо перезапустить службу без обрыва соединений:

```
sudo systemctl reload angie.service
```

6.4.3.6 Настройка обратного проксирования в Angie

Веб-сервер Angie, как и Nginx, может использоваться в качестве обратного прокси (reverse proxy) для передачи запросов к другим серверам приложений, например: Apache, Tomcat, Node.js и др. Данный механизм позволяет централизовать обработку SSL, балансировать нагрузку и распределять трафик.

Для настройки обратного проксирования необходимо создать блок server, в котором используется директива `proxy_pass`. Ниже приведён минимальный пример:

```
server {
    listen       80;
    server_name  proxy.example.local;

    location / {
        proxy_pass http://127.0.0.1:8080;
```

```
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header                    X-Forwarded-For
$proxy_add_x_forwarded_for;
        }
    }
```

В данной конфигурации все запросы, поступающие на порт 80, перенаправляются на внутренний сервер по адресу 127.0.0.1:8080. Дополнительно устанавливаются заголовки для передачи исходного IP-адреса клиента и имени хоста.

Балансировка нагрузки

Angie поддерживает балансировку между несколькими серверами приложений с помощью блока upstream, например:

```
    upstream backend {
        server 192.168.1.10 max_fails=3 fail_timeout=30s;
        server 192.168.1.11 max_fails=3 fail_timeout=30s;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
        }
    }
```

В данной конфигурации создаётся пул из двух серверов с балансировкой по умолчанию (round-robin) и базовым механизмом проверки доступности через параметры max_fails и fail_timeout.

Привязка сессий

Для случаев, когда требуется сохранять сессию пользователя на одном и том же бэкенде, используется механизм привязки сессий (sticky sessions), например:

```
    upstream backend {
        server 192.168.1.10;
        server 192.168.1.11;
        sticky cookie srv_id expires=1h domain=.example.local
path=/;
```

```
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
        }
    }
```

При данной настройке создаётся cookie srv_id, позволяющее направлять все запросы пользователя в течение часа на один и тот же сервер.

Тайм-ауты и ограничения

Рекомендуется дополнительно настроить параметры тайм-аутов, чтобы защитить сервер от зависших соединений:

```
    proxy_connect_timeout       5s;
    proxy_send_timeout         30s;
    proxy_read_timeout         30s;
    send_timeout               30s;
```

Также могут использоваться лимиты на количество соединений с одного адреса или ограничение скорости:

```
    limit_conn_zone $binary_remote_addr zone=addr:10m;
    limit_conn addr 10;

    limit_req_zone $binary_remote_addr zone=req:10m rate=1r/s;
    limit_req zone=req burst=5;
```

После внесения всех изменений в конфигурационные файлы необходимо выполнить проверку синтаксиса:

```
    sudo angie -t
```

После проверки необходимо перезапустить службу без обрыва соединений:

```
    sudo systemctl reload angie.service
```

6.4.3.7 Ограничения и защита в Angie

В целях повышения безопасности и устойчивости веб-сервера Angie рекомендуется настроить ряд механизмов ограничения ресурсов и фильтрации запросов. Это позволяет минимизировать риск DoS-атак, защитить сервис от чрезмерной нагрузки и контролировать активность пользователей:

– Ограничение количества подключений от одного IP-адреса директивой `limit_conn`:

```
    limit_conn_zone $binary_remote_addr zone=addr:10m;
    limit_conn addr 20;
```

где для IP-адреса создаётся зона размером 10 МБ и допускается не более 20 одновременных подключений.

– Ограничение скорости передачи данных к клиентам с помощью `limit_rate`:

```
    server {
        location / {
            limit_rate 500k;
        }
    }
```

где задан параметр limit_rate 500k, который позволит не превышать скорость 500 КБ в секунду для одного соединения.

– Ограничение скорости запросов с помощью `limit_req`:

```
    limit_req_zone $binary_remote_addr zone=req_zone:10m
rate=5r/s;
    limit_req zone=req_zone burst=10 nodelay;
```

где задается не более 5 запросов в секунду на один IP-адрес с возможностью коротких всплесков до 10 запросов подряд.

– Для фильтрации доступа к ресурсу по IP-адресам можно использовать директивы `allow` и `deny`:

```
    server {
        location / {
            allow 192.168.1.0/24;
            deny all;
        }
    }
```

В данном случае доступ будет разрешён только из подсети 192.168.1.0/24, остальные подключения будут отклоняться.

– Защита от медленных атак типа slowloris с помощью настройки тайм-аутов:

```
    client_body_timeout 10s;
    client_header_timeout 10s;
```

```
    send_timeout 30s;
    keepalive_timeout 15s;
```

Эти параметры ограничивают время ожидания данных от клиента, уменьшая вероятность блокировки рабочих процессов медленными соединениями.

– Для дополнительной защиты возможно использование fail2ban, который анализирует журналы ошибок Angie и блокирует IP-адреса с подозрительной активностью. Рекомендуется настроить fail2ban с фильтром для error.log и указанием параметров бана в файле /etc/fail2ban/jail.conf или отдельном jail-файле, например:

```
    [angie]
    enabled  = true
    filter   = angie
    port     = http,https
    logpath  = /var/log/angie/error.log
    maxretry = 5
    findtime = 600
    bantime  = 3600
```

6.4.3.8 Логирование и ротация журналов в Angie

Веб-сервер Angie обладает гибкими возможностями по ведению журналов запросов и ошибок, которые позволяют контролировать его работу, проводить аудит и своевременно реагировать на возникающие сбои или аномалии.

По умолчанию Angie сохраняет журналы обращений в файл /var/log/angie/access.log, а ошибки – в файл /var/log/angie/error.log. Форматы записей можно изменять в соответствии с требованиями эксплуатации, например, добавив в них дополнительную информацию о заголовках, времени ответа, реферере или пользовательском агенте. Для этого используется директива `log_format`. Пример расширенного шаблона:

```
    log_format  main  '$remote_addr - $remote_user [$time_local]
"$request" '
                      '$status $body_bytes_sent "$http_referer"
'
                      '"$http_user_agent"
"$http_x_forwarded_for"';
    access_log  /var/log/angie/access.log  main;
```

Уровень детализации сообщений об ошибках регулируется директивой `error_log`. Например, для вывода предупреждений и ошибок достаточно указать уровень warn:

```
    error_log  /var/log/angie/error.log  warn;
```

При эксплуатации нескольких виртуальных хостов рекомендуется организовать отдельные журналы для каждого сайта, что позволит быстрее анализировать проблемы и распределять нагрузку при аудите, например:

```
    server {
        server_name site1.local;
        access_log /var/log/angie/site1_access.log;
        error_log  /var/log/angie/site1_error.log warn;
    }
```

Также Angie поддерживает вывод журналов через системный журнал journald или через syslog-сервер. Для этого в настройках `access_log` и `error_log` можно указать:

```
    access_log
syslog:server=unix:/dev/log,facility=local7,tag=angie_access;
    error_log
syslog:server=unix:/dev/log,facility=local7,tag=angie_error warn;
```

Это обеспечивает централизованный сбор журналов для последующего анализа средствами системного журнала.

Ротация журналов

Для управления размерами файлов журналов и автоматического их сжатия рекомендуется применять системную утилиту `logrotate`. При стандартной установке создаётся конфигурация в каталоге /etc/logrotate.d/angie, которая может быть настроена, например, следующим образом:

```
    /var/log/angie/*.log {
        daily
        missingok
        rotate 7
        compress
        delaycompress
        notifempty
        create 0640 nginx adm
        sharedscripts
        postrotate
```

```
            sudo systemctl reload angie.service > /dev/null 2>&1
|| true
        endscript
    }
```

В приведённой конфигурации ротация выполняется ежедневно с хранением семи предыдущих архивов, причём старые логи сжимаются для экономии места. После ротации автоматически отправляется команда перезагрузки конфигурации сервера, чтобы Angie корректно переключился на новый лог-файл.

6.4.3.9 Мониторинг и диагностика в Angie

Для эффективного администрирования веб-сервера Angie необходимо обеспечить постоянный мониторинг его состояния и использование встроенных инструментов диагностики.

Angie предоставляет расширенные возможности мониторинга за счёт встроенного REST API и визуальной консоли Console Light. REST API позволяет в реальном времени получать сведения о конфигурации, состоянии клиентских соединений, статистике зон разделяемой памяти, а также о параметрах работы проксируемых серверов. Данные возвращаются в формате JSON, что упрощает интеграцию с внешними системами мониторинга, например, Zabbix или Prometheus.

Для Prometheus в Angie реализован собственный экспорт метрик с возможностью настройки шаблонов. Это позволяет агрегировать показатели нагрузки и работы сервера в стандартных панелях мониторинга, а также формировать предупреждения при выходе параметров за заданные пределы.

Кроме того, Angie поддерживает визуальную консоль Console Light, которая позволяет в браузере наблюдать за ключевыми показателями веб-сервера. Console Light не требует отдельной установки и доступна сразу после включения в конфигурации. Для её активации достаточно добавить в основной конфигурационный файл соответствующий блок и открыть указанный интерфейс по адресу сервера.

Мониторинг и диагностика могут дополняться средствами анализа логов. Рекомендуется:

– отслеживать изменения в журналах ошибок и доступа с помощью системного журнала или `logrotate`;

– контролировать ключевые тайм-ауты и показатели использования памяти;

– проверять своевременность применения обновлений конфигурации с помощью команды `angie -t`;

– анализировать производительность с помощью отчётов Prometheus или других инструментов.

6.4.3.10 Резервное копирование и восстановление конфигурации Angie

Для обеспечения надёжной и стабильной работы веб-сервера Angie важно организовать корректное резервное копирование его конфигурационных файлов, а также предусмотреть процедуры быстрого восстановления при сбоях или ошибках конфигурации.

В первую очередь необходимо включить в резервную копию следующие элементы:

– Основной файл конфигурации /etc/angie/angie.conf.
– Все файлы виртуальных хостов, например, в каталогах /etc/angie/conf.d/, /etc/angie/sites-available/ и /etc/angie/sites-enabled/.
– Пользовательские сертификаты и ключи в каталогах /etc/ssl/certs/ и /etc/ssl/private/.
– Файлы журналов конфигурационных шаблонов мониторинга (если используются).

Рекомендуется использовать утилиты `tar` или `rsync`, позволяющие быстро создать резервную копию всей структуры конфигурации вместе с правами доступа (подробнее о работе с утилитами рассмотрено в разделе 7.2). Пример команды:

```
sudo tar czf /backup/angie-config-$(date +%Y%m%d).tar.gz /etc/angie /etc/ssl
```

Архив, созданный таким образом, можно переместить на внешний носитель или в защищённое хранилище.

При необходимости восстановления достаточно выполнить распаковку архива в исходное место и проверить корректность прав доступа:

```
sudo tar xzf angie-config-20250620.tar.gz -C /
```

После восстановления рекомендуется:

– убедиться в целостности всех файлов конфигурации;
– проверить синтаксис конфигурации командой `angie -t`;
– выполнить перезапуск или перезагрузку конфигурации Angie:

```
sudo systemctl reload angie.service
```

Для повышения надёжности рекомендуется дополнительно вести журнал изменений конфигурации, фиксируя дату, причину и перечень изменений. Это позволит избежать ошибок при откате и быстрее выявлять некорректные правки.

Также следует рассмотреть возможность интеграции конфигурационных файлов Angie в систему управления версиями (например, Git), что обеспечит контроль версий и удобное отслеживание изменений между разными этапами эксплуатации.

Таким образом, регулярное резервное копирование и чётко прописанные процедуры восстановления позволяют минимизировать риски при эксплуатации веб-сервера Angie и быстро возвращать его к работоспособному состоянию в случае сбоев.

6.4.3.11 Миграция с Nginx на Angie

Поскольку Angie является ответвлением Nginx и сохраняет полную совместимость с его конфигурационным синтаксисом, процесс миграции обычно не вызывает значительных трудностей и может быть выполнен с минимальными изменениями.

Angie поддерживает все основные директивы и модули, используемые в Nginx, включая виртуальные хосты, обратное проксирование, балансировку нагрузки, TLS/SSL, фильтры, а также возможности журналирования. Поэтому в большинстве случаев достаточно скопировать существующие файлы конфигурации Nginx в директории конфигурации Angie.

При выполнении миграции с Nginx на Angie рекомендуется придерживаться следующего порядка:

а) Подготовка и анализ конфигурации:

– проверка актуальности версий Nginx и Angie;
– проверка установленных модулей Nginx на неиспользование закрытых проприетарных расширений, отсутствующих в Angie;
– сохранить резервную копию всех конфигурационных файлов Nginx.

б) Установка Angie:

– установка пакета Angie через менеджер пакетов `dnf`;
– проверка доступности службы и наличия необходимых зависимостей;
– отключение службу Nginx для исключения конфликта портов.

в) Перенос конфигурации:

– копирование данных из `/etc/nginx` в `/etc/angie`;
– проверка всех include-директив, которые должны указывать корректные пути;

– проверка корректности путей к логам и каталогам виртуальных хостов.

г) Тестирование:

– проверка синтаксис конфигурации Angie командой `angie -t`;

– локальное тестирование на стенде для проверки работоспособности всех виртуальных хостов, TLS, проксирования и других функций;

– при необходимости корректировка параметров, добавление новых возможностей Angie (например, `http3`, `slow_start`).

д) Ввод в эксплуатацию:

– запуск службу Angie;

– проверка корректности функционирования всех сайтов и сервисов;

– настройка мониторинга и журналирование для оперативного контроля состояния.

е) Рекомендации для выполнения успешной миграции:

– обязательное сохранение резервных копий всех конфигурационных файлов до начала миграции;

– тестирование работы новых функций Angie сначала в тестовом окружении;

– при работе с динамическими модулями проверка их совместимости;

– ведение журнала всех действий по миграции и фиксирование возможных проблем для последующего анализа.

6.5 Веб-интерфейс управления Cockpit

Cockpit представляет собой инструмент для администрирования устройств под управлением ОС РОСА "ХРОМ" и других ОС на основе Linux, основанный на удобном веб-интерфейсе. Cockpit позволяет выполнять основные задачи системного администрирования с помощью любого современного веб-браузера, обеспечивая простое и централизованное управление сервером без необходимости подключаться к нему через консоль напрямую.

С помощью Cockpit можно:

– просматривать состояние ресурсов сервера (процессора, памяти, дисков);
– управлять службами и процессами;
– контролировать состояние сети и настраивать сетевые интерфейсы;
– отслеживать журналы системы;
– управлять пользователями и группами;
– обновлять пакеты;
– подключаться к консоли системы прямо из браузера.

Cockpit может служить важным элементом инфраструктуры администрирования, особенно в распределённых или изолированных средах.

Для установки Cockpit вместе с пакетом расширенного мониторинга PCP следует выполнить команду:

```
sudo dnf install cockpit cockpit-pcp
```

После установки необходимо активировать сокет Cockpit, чтобы сервис запускался при старте Системы:

```
sudo systemctl enable cockpit.socket
```

Для немедленного запуска службы используется команда:

```
sudo systemctl start cockpit.socket
```

После запуска веб-интерфейс Cockpit будет доступен по адресу:

```
http://<ip-адрес сервера>:9090/
```

где <ip-адрес сервера> – это IP-адрес или полное доменное имя хоста, на котором установлен Cockpit.

Cockpit поддерживает установку дополнительных модулей, расширяющих его возможности. Чтобы получить список доступных модулей в репозитории, используется команда:

```
dnf search cockpit-
```

Основные модули Cockpit:

– cockpit-machines – управление виртуальными машинами (libvirt/KVM), включая создание, удаление и настройку ВМ;

– cockpit-storaged – управление дисками, разделами, томами LVM и RAID-массивами;

– cockpit-networkmanager – удобное управление сетевыми интерфейсами через NetworkManager;

– cockpit-sosreport – сбор отчётов о состоянии Системы для последующего анализа;

– cockpit-podman – работа с контейнерами Podman, включая их запуск и мониторинг;

– cockpit-kubernetes – управление кластерами Kubernetes и отслеживание их состояния;

– cockpit-packagekit – обновление и установка программного обеспечения через интерфейс PackageKit;

– cockpit-selinux – просмотр и базовое управление политиками SELinux;

– cockpit-composer – создание и генерация собственных ISO-образов и готовых для развёртывания систем;

– cockpit-pcp – расширенный мониторинг производительности с помощью Performance Co-Pilot (PCP);

– cockpit-navigator – интеграция с командной строкой в стиле Midnight Commander для удобного управления файлами.

Для установки необходимого модуля нужно выполнить:

```
sudo dnf install cockpit-<модуль>
```

где <модуль> – это имя выбранного расширения.

После установки Cockpit рекомендуется:

– ограничить доступ к порту 9090 только с доверенных IP-адресов через брандмауэр;

– при необходимости настроить HTTPS для защиты передаваемых данных, указав путь к TLS-сертификату и приватному ключу в конфигурации Cockpit;

– ограничить список пользователей, имеющих право входа, через стандартные механизмы контроля групп (например, группа wheel).

6.6 Сервер журналирования

6.6.1 journald

6.6.1.1 Общие сведения

Централизованный сервер журналирования может помочь с анализом событий на компьютерах в ИТ-инфраструктуре, объединенной в систему. Настроить их сбор с журналов можно с помощью journald.

В ОС используется сервис journald. Journald входит в состав systemd.

В systemd предусмотрены специальные компоненты для решения задачи логирования: systemd-journal-remote, systemd-journal-upload и systemd-journal-gatewayd.

6.6.1.2 Установка сервера

Для работы нужно установить пакет systemd-journal-gateway на всех трех хостах, перечисленных ниже, командой:

```
sudo dnf install systemd-journal-gateway
```

В качестве примера в пп.6.6.1.3-6.6.1.6 будет использовано три разных хоста:

– rosaserver – сервер логирования journald;

– rosaserver1 – первый клиент (сервер/десктоп);
– rosaserver2 – второй клиент (сервер/десктоп).

6.6.1.3 Настройка сервера

Настройка состоит из двух частей. В первой части настраивается сам сервер с именем хоста rosaserver.

Во второй части настраиваются два клиента (сервера/десктопа) с именами хостов/IP: rosaserver1/192.168.1.111 и rosaserver2/192.168.1.232.

В первую очередь осуществляется настройка systemd-journal-remote на сервере. В ходе настройки необходимо совершить несколько шагов:

а) Прописать в файле /etc/hosts клиентов:

```
192.168.1.111 rosaserver1
192.168.1.232 rosaserver2
```

б) Создать директорию /var/log/journal/remote, в которой будут храниться все журналы с клиентов:

```
mkdir -p /var/log/journal/remote
```

в) Предоставить права на эту директорию:

```
chown systemd-journal-remote:systemd-journal-remote /var/log/journal/remote
```

Для дальнейшей настройки сервера следует выполнить следующие действия:

а) Создать директорию для сертификата:

```
sudo mkdir /etc/pki/tls/ca
sudo ln -s /etc/pki/tls/ca /etc/ssl/ca
```

Остальные директории /etc/ssl/certs и /etc/ssl/private уже созданы.

б) Сгенерировать ключ сервера:

```
openssl req -newkey rsa:2048 -days 3650 -x509 -nodes -out /etc/ssl/ca/caserver.pem -keyout /etc/ssl/ca/caserver.key -subj '/CN=Certificate authority/'
```

в) Сгенерировать сертификаты для сервера (опция CN-SERVER должна совпадать с именем хоста сервера; в данном случае /CN=rosaserver):

```
openssl req -newkey rsa:2048 -nodes -out /etc/ssl/certs/rosaserver.csr -keyout /etc/ssl/certs/rosaserver.key -subj "/CN=rosaserver/"
```

```
    openssl x509 -req -in /etc/ssl/certs/rosaserver.csr -CA
/etc/ssl/ca/caserver.pem -CAkey /etc/ssl/ca/caserver.key -out
/etc/ssl/private/rosaserver.pem
```

г) Сгенерировать сертификаты для первого клиента (опция CN-CLIENT должна совпадать с именем хоста клиента, в данном случае /CN=rosaserver1):

```
    openssl req -newkey rsa:2048 -nodes -out
/etc/ssl/certs/rosaserver1.csr -keyout
/etc/ssl/certs/rosaserver1.key -subj "/CN=rosaserver1/"
    openssl x509 -req -in /etc/ssl/certs/rosaserver1.csr -CA
/etc/ssl/ca/caserver.pem -CAkey /etc/ssl/ca/caserver.key -out
/etc/ssl/private/rosaserver1.pem
```

д) Генерировать сертификаты для второго клиента (опция CN-CLIENT должна совпадать с именем хоста клиента, в данном случае /CN=rosaserver2):

```
    openssl req -newkey rsa:2048 -nodes -out
/etc/ssl/certs/rosaserver2.csr -keyout
/etc/ssl/certs/rosaserver2.key -subj "/CN=rosaserver2/"
    openssl x509 -req -in /etc/ssl/certs/rosaserver2.csr -CA
/etc/ssl/ca/caserver.pem -CAkey /etc/ssl/ca/caserver.key -out
/etc/ssl/private/rosaserver2.pem
```

е) Сгенерированные ключи и сертификаты для сервера уже размещены в нужно месте. Требуется скопировать ключи и сертификаты для клиентов.

ж) На сервере выдать нужные права:

```
    chmod 644 /etc/ssl/certs/rosaserver.key
    chmod 644 /etc/ssl/private/rosaserver.pem
```

з) Отредактировать файл конфигурации /etc/systemd/journal-remote.conf к следующему виду:

```
    [Remote]
    Seal=false
    SplitMode=host
    ServerKeyFile=/etc/ssl/certs/rosaserver.key
    ServerCertificateFile=/etc/ssl/private/rosaserver.pem
    TrustedCertificateFile=/etc/ssl/ca/caserver.pem
```

и) Сохранить файл и выйти.

6.6.1.4 Настройка клиента

Для настройки клиентов нужно выполнить следующие действия:

а) Прописать сервер в файле /etc/hosts на каждом из клиентов:

```
192.168.1.15 rosaserver
```

б) Создать директорию для сертификата:

```
sudo mkdir /etc/pki/tls/ca
sudo ln -s /etc/pki/tls/ca /etc/ssl/ca
```

Остальные директории и ссылки /etc/ssl/certs и /etc/ssl/private уже созданы.

в) Войти на сервер и скопировать ключи и сертификаты для первого клиента по SSH:

```
scp /etc/ssl/ca/caserver.pem root@192.168.1.111:/etc/ssl/ca/
scp                          /etc/ssl/private/rosaserver1.pem
root@192.168.1.111:/etc/ssl/private/
scp                            /etc/ssl/certs/rosaserver1.key
root@192.168.1.111:/etc/ssl/certs/
```

г) Скопировать ключи и сертификаты для второго клиента по SSH:

```
scp /etc/ssl/ca/caserver.pem root@192.168.1.232:/etc/ssl/ca/
scp                          /etc/ssl/private/rosaserver2.pem
root@192.168.1.232:/etc/ssl/private/
scp                            /etc/ssl/certs/rosaserver2.key
root@192.168.1.232:/etc/ssl/certs/
```

д) Предоставить нужные права на первом клиенте:

```
chmod 644 /etc/ssl/certs/rosaserver1.key
chmod 644 /etc/ssl/private/rosaserver1.pem
```

е) Предоставить нужные права на втором клиенте:

```
chmod 644 /etc/ssl/certs/rosaserver2.key
chmod 644 /etc/ssl/private/rosaserver2.pem
```

ж) Привести файл конфигурации первого клиента к виду:

```
[Upload]
URL=https://rosaserver:19532
ServerKeyFile=/etc/ssl/certs/rosaserver1.key
ServerCertificateFile=/etc/ssl/private/rosaserver1.pem
TrustedCertificateFile=/etc/ssl/ca/caserver.pem
```

з) Привести файл конфигурации второго клиента к виду:

```
[Upload]
```

```
URL=https://rosaserver:19532
ServerKeyFile=/etc/ssl/certs/rosaserver2.key
ServerCertificateFile=/etc/ssl/private/rosaserver2.pem
TrustedCertificateFile=/etc/ssl/ca/caserver.pem
```

6.6.1.5 Запуск логирования journald

Для запуска логирования нужно выполнить запуск:

а) на сервере:

```
sudo systemctl start systemd-journal-gateway.service
sudo systemctl start systemd-journal-remote.service
```

б) на всех клиентах:

```
sudo systemctl start systemd-journal-gateway.service
sudo systemctl start systemd-journal-upload.service
```

6.6.1.6 Проверка работоспособности

Для чтения собственных логов самого сервера используется команда:

```
journalctl
```

Для чтения логов с удаленных клиентов используется команда:

```
sudo journalctl -D /var/log/journal/remote
```

Для проверки лога с удаленного сервера необходимо выполнить следующие шаги:

а) Войти на второй клиент, (на rosaserver2) и выполнить команду:

```
logger -p syslog.debug "### TEST-2 MESSAGE from client rosaserver2 ###"
```

б) Войти на сервер и посмотреть, как данная запись с удаленного клиента rosaserver2 отразилась в журнале на сервере, с помощью команды:

```
sudo journalctl -D /var/log/journal/remote/ | grep TEST
```

Примерный результат приведен ниже (рисунок 106).

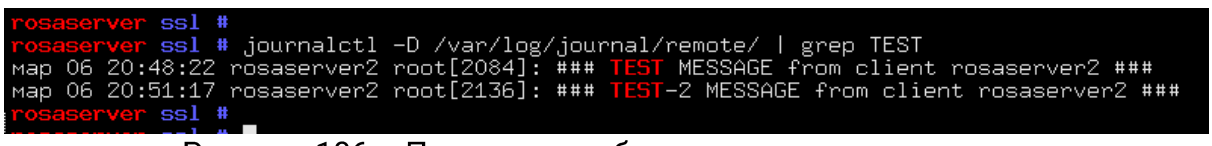


Рисунок 106 – Проверка отображения записи с сервера

Логи с удаленных серверов будут храниться в директории /var/log/journal/remote по имени хоста (рисунок 107):

```
rosaserver ssl #
rosaserver ssl # ls -al /var/log/journal/remote/
итого 16408
drwxr-xr-x+ 2 systemd-journal-remote systemd-journal-remote    4096 мар  6 19:48  .
drwxr-sr-x+ 5 root                   systemd-journal           4096 мар  6 16:06  ..
-rw-r-----+ 1 systemd-journal-remote systemd-journal-remote 8388608 мар  6 20:21 'remote-CN=rosaserver1.journal'
-rw-r-----+ 1 systemd-journal-remote systemd-journal-remote 8388608 мар  6 20:48 'remote-CN=rosaserver2.journal'
rosaserver ssl #
```

Рисунок 107 – Хранение логов удаленных серверов

6.6.2 rsyslog

6.6.2.1 Общие сведения

Rsyslog является службой системного журналирования, предназначенной для получения, обработки, хранения и передачи системных сообщений. Служба поддерживает работу как локального, так и удалённого журналирования, обеспечивая централизованный сбор логов. Rsyslog используется совместно с systemd-journald.

Служба rsyslog обеспечивает расширенные функции обработки логов, включая:

– приём сообщений от системных служб, приложений и ядра ОС;

– фильтрацию логов по уровням важности, категориям, программам и другим критериям;

– маршрутизацию сообщений в различные файлы, каталоги или сетевые направления;

– поддержку удалённого журналирования по протоколам UDP и TCP;

– запись журналов в структурированном виде и поддержку кастомных форматов вывода;

– интеграцию с journald для обработки системных сообщений;

– возможность масштабирования при большом объёме журналов.

Rsyslog работает совместно с journald: journald передаёт сообщения в rsyslog через сокет /run/systemd/journal/syslog.

Rsyslog рекомендуется использовать в следующих случаях:

– централизованный сбор журналов ото всех узлов к одному серверу для последующего анализа;

– организация долгосрочного хранения логов;

– передача сообщений служб безопасности на выделенный сервер;

– разделение журналов по хостам, службам или уровням важности;

– интеграция журналирования в комплексные системы мониторинга и аудита.

Rsyslog поддерживает режим проверки синтаксиса с помощью команды:

```
rsyslogd -N1
```

При наличии ошибок команда вернёт описание проблемных строк.

6.6.2.2 Конфигурация и синтаксис rsyslog

Основной конфигурационный файл службы расположен по пути /etc/rsyslog.conf.

Данный файл содержит базовые настройки, параметры загрузки модулей, глобальные параметры работы системы и общие правила обработки сообщений. Изменение конфигурационного файла допускается только пользователем с административными правами.

Дополнительные правила размещаются в каталоге /etc/rsyslog.d/. Каждый файл с расширением .conf в данном каталоге автоматически применяется службой при её запуске. Таким образом можно создавать отдельные файлы для индивидуальных служб или групп узлов.

Изменение файлов в каталоге /etc/rsyslog.d/ рекомендуется выполнять вместо изменения основного файла /etc/rsyslog.conf, чтобы избежать конфликтов при обновлениях Системы.

Конфигурация rsyslog исполняется в следующей последовательности:

а) модули ввода (input modules) — определяют источники сообщений (например, приём сообщений по UDP или TCP);

б) фильтры (filters) — определяют условия обработки сообщений (выбор сообщений по уровню важности, имени хоста, имени программы);

в) действия (actions) — определяют, куда передаётся сообщение (запись в файл, отправка на удалённый сервер).

Основной конфигурационный файл включает следующие элементы:

– загрузка необходимых модулей;
– включение поддержки удалённого приёма сообщений (опционально);
– определение формата сообщений;
– обработка сообщений journald;
– правила маршрутизации и фильтрации;
– указание расположения файлов журналов.

Пример минимального конфигурационного фрагмента:

```
module(load="imuxsock")     # получение сообщений от
локальных служб
```

```
    module(load="imjournal")    # интеграция с journald


    *.info;mail.none;authpriv.none;cron.none
/var/log/messages
    authpriv.*                                  /var/log/secure
```

Служба rsyslog поддерживает два формата записи конфигурационных правил: классический синтаксис (наследуемый от syslog) и современный синтаксис RainerScript. Оба формата могут использоваться совместно в рамках одной конфигурации.

Классический синтаксис представляет собой краткий формат записи правил маршрутизации сообщений, применяемый в традиционных реализациях syslog. Правило таких конфигураций имеет вид:

```
    facility.level    destination
```

где:

– facility — категория сообщений (authpriv, cron, mail, daemon и др.);
– level — уровень важности сообщения;
– destination — путь к файлу или иной канал вывода.

Например:

```
    authpriv.*    /var/log/secure
    *.info        /var/log/messages
```

Современный синтаксис основан на параметрическом описании операций и позволяет формировать более сложные правила маршрутизации и фильтрации. Конфигурация включает загрузку модулей, параметры вводимых потоков, блоки условий и действия:

```
    module(load="имя_модуля")
    input(type="тип_ввода" параметр="значение")


    if <условие> then <действие>
```

Например:

```
    module(load="imudp")
    input(type="imudp" port="514")


    if $syslogseverity-text == "err" then /var/log/errors.log
```

6.6.2.3 Настройка rsyslog как сервера журналирования

Для использования узла как централизованного сервера журналирования необходимо включить поддержку входящих сообщений. 514 — это порт, на котором слушает rsyslog.

Пример активации модуля UDP (порт 514):

```
module(load="imudp")
input(type="imudp" port="514")
```

Пример активации модуля TCP:

```
module(load="imtcp")
input(type="imtcp" port="514")
```

После внесения изменений служба rsyslog должна быть перезапущена:

```
systemctl restart rsyslog
```

Для проверки того, что порты открыты, используется команда:

```
ss -tulnp | grep 514
```

Рекомендуется сохранять журналы каждого удалённого узла в отдельный каталог. Типовой подход — использование динамического имени хоста источника, который настраивается командой:

```
$template
RemoteLogs,"/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log"

*.* ?RemoteLogs
```

Такой шаблон создаёт структуру:

```
/var/log/remote/
    ├── host1/
    │     ├── sshd.log
    │     ├── cron.log
    │     └── messages.log
    ├── host2/
    │     └── ...
```

Создание каталога выполняется автоматически при появлении первых журналов.

Для проверки приёма сообщений на сервере используется утилита logger на удалённом хосте:

```
logger "test message from client"
```

Сообщение должно появиться в журнале сервера, например:

```
/var/log/remote/<имя_клиента>/messages.log
```

Если сообщение отсутствует, следует проверить:

– работу службы rsyslog на сервере и клиенте;
– доступность портов;
– правильность указания IP-адреса сервера;
– настройки фильтрации в конфигурации.

6.6.2.4 Настройка rsyslog как клиента

Отправка логов на удалённый сервер

На клиентском узле необходимо указать адрес сервера журналирования и способ передачи сообщений.

Отправка по UDP настраивается командой:

```
*.* @192.168.1.10:514
```

Отправка по TCP настраивается командой:

```
*.* @@192.168.1.10:514
```

Один символ "@" обозначает UDP, два символа "@@" — TCP.

Запись следует размещать в файле:

```
/etc/rsyslog.d/remote.conf
```

Пересылка всех сообщений

Для передачи всех системных сообщений используется команда:

```
*.* @@192.168.1.10:514
```

После изменения конфигурации необходимо перезагрузить сервис:

```
systemctl restart rsyslog
```

Проверка отправки сообщений выполняется с помощью команды:

```
logger "Message from test client"
```

Сообщение "Message from test client" должно появиться на сервере.

Селективная пересылка сообщений

Пересылка только событий уровня error и выше:

```
*.err @@192.168.1.10:514
```

Пересылка только событий от SSH:

```
if $programname == "sshd" then @@192.168.1.10:514
```

Исключение определённых служб:

```
if $programname == "cron" then stop
```

Селективная пересылка снижает нагрузку на канал и сервер журналирования.

Проверка работы клиента осуществляется с помощью команды:

```
logger -p user.info "Client test log"
```

На сервере сообщение должно появиться в каталоге удалённого хоста. При отсутствии сообщений необходимо проверить:

– сетевую доступность порта 514;
– включена ли отправка (UDP/TCP);
– корректность синтаксиса конфигурации клиента;
– ошибки службы rsyslog (`journalctl -u rsyslog`).

6.6.2.5 Передача сообщений journald в rsyslog

Для передачи сообщений journald в rsyslog используется сокет:

```
/run/systemd/journal/syslog
```

При загрузке службы rsyslog модуль imjournal обеспечивает приём сообщений journald:

```
module(load="imjournal")
```

Передаваемые journald сообщения затем обрабатываются rsyslog в соответствии с заданными правилами. Таким образом достигается совместная работа обеих подсистем: journald осуществляет сбор, а rsyslog — фильтрацию, хранение и отправку.

6.6.2.6 Диагностика работы rsyslog

Для просмотра состояния службы используется команда:

```
journalctl -u rsyslog
```

Команда показывает ошибки загрузки модулей, проблемы сети и нарушения синтаксиса конфигурации.

Для детализированной диагностики возможно включение отладочного режима:

```
rsyslogd -dn
```

Режим выводит последовательность обработки сообщений и помогает выявить ошибки маршрутизации.

Файлы логов, создаваемые rsyslog, должны обслуживаться утилитой logrotate.

Пример файла /etc/logrotate.d/rsyslog-remote:

```
/var/log/remote/*/*.log {
    daily
    rotate 14
    compress
    missingok
    notifempty
}
```

Для ограничения размеров логов рекомендуется:

– включать сжатие старых логов;
– ограничивать количество хранимых файлов;
– выделять отдельный раздел для /var/log/remote.

6.7 Сервер печати CUPS

Сервер печати в ОС основан на системе CUPS (Common UNIX Printing System), которая является стандартным решением для управления печатью в UNIX-подобных системах. CUPS поддерживает различные принтеры и позволяет организовать печать как для локальных, так и для сетевых пользователей.

Настройка сервера печати может потребоваться в сетевой среде, где необходимо централизованное управление печатью для нескольких клиентов.

Сервер CUPS уже установлен в Системе, но если он отсутствует, его можно установить с помощью команды:

```
sudo dnf install cups
```

После установки необходимо запустить службу CUPS и добавить её в автозагрузку:

```
sudo systemctl start cups
sudo systemctl enable cups
```

Если доступ к серверу CUPS ограничен только локальной машиной, то для настройки доступа других клиентов к серверу печати, необходимо внести изменения в конфигурационный файл /etc/cups/cupsd.conf.

Для редактирования конфигурационного файла /etc/cups/cupsd.conf нужно:

1) открыть файл в текстовом редакторе:

```
sudo nano /etc/cups/cupsd.conf
```

2) найти и изменить параметры в блоках Listen и Allow для разрешения удаленного доступа:

```
Listen *:631
<Location />
Order allow,deny
Allow all
</Location>
```

3) после изменений сохранить файл и перезапустить службу CUPS:

```
sudo systemctl restart cups
```

6.7.1 Добавление и настройка принтеров

Для добавления принтера может использоваться веб-интерфейс CUPS, который доступен по адресу http://localhost:631. Этот интерфейс позволяет управлять принтерами, просматривать очередь печати и настраивать параметры сервера (рисунок 108).

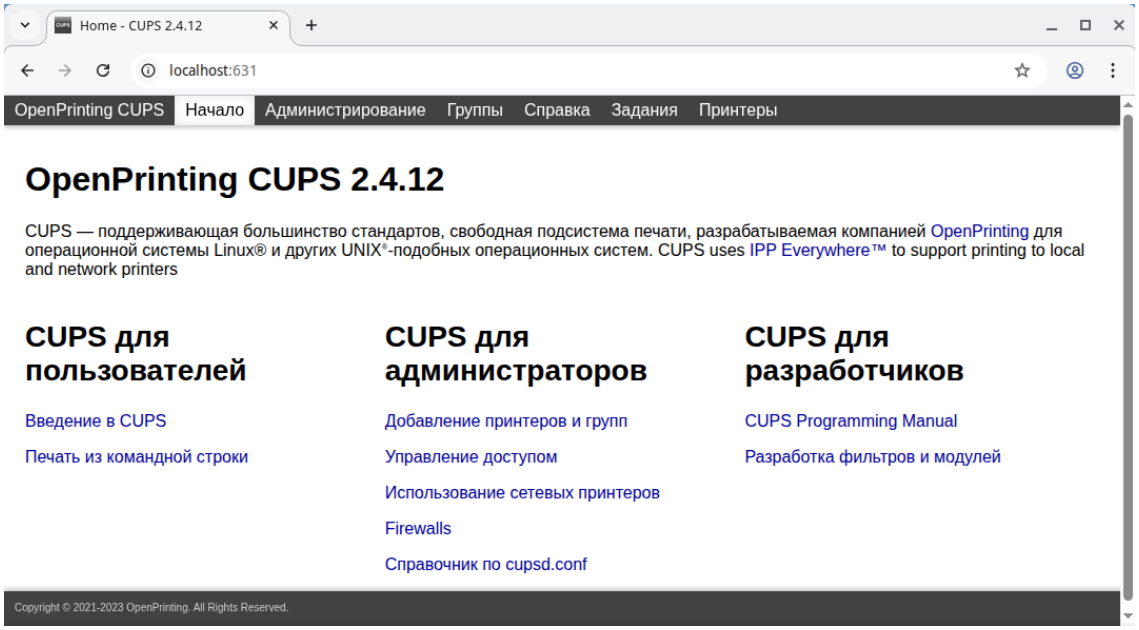


Рисунок 108 – Веб-интерфейс управления сервером CUPS

Для добавления принтера через веб-интерфейс CUPS необходимо:

а) Перейти в веб-интерфейс, затем – в раздел "Администрирование" (рисунок 109). Для входа необходимо ввести логин и пароль учетной записи Системы.

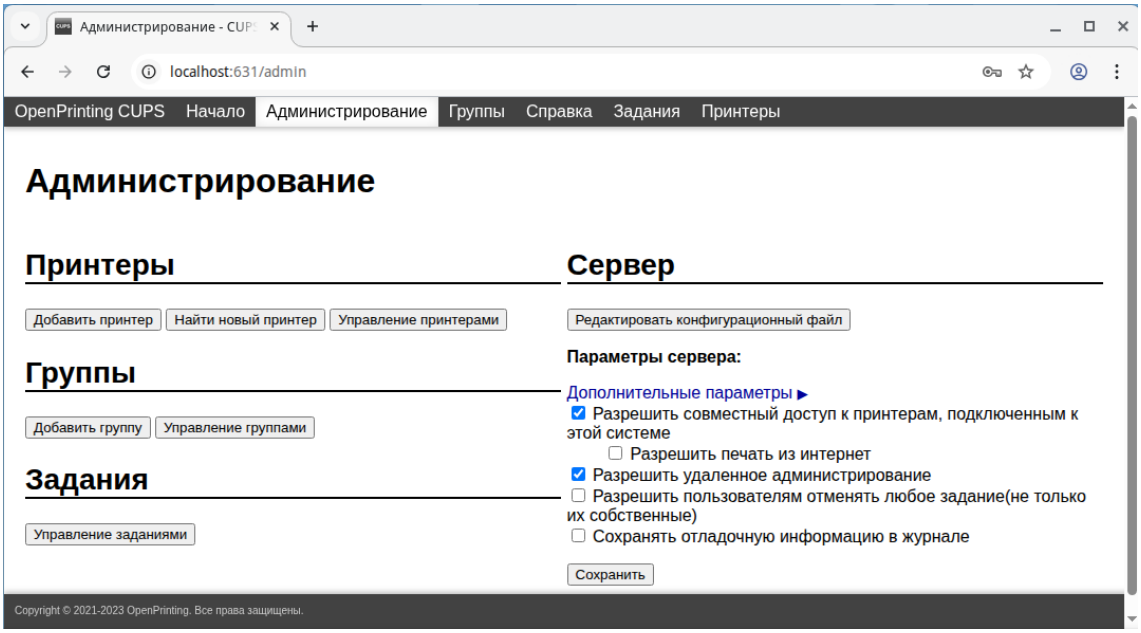


Рисунок 109 – Панель администрирования веб-интерфейса CUPS

б) Нажать кнопку `Добавить принтер` и следовать инструкциям мастера настройки.

В большинстве случаев CUPS автоматически определяет модель принтера и подбирает подходящий драйвер. Если нужный принтер не поддерживается автоматически, необходимо выбрать драйвер вручную или установить его из дополнительных пакетов.

Если принтер больше не используется, его можно удалить через веб-интерфейс или команду:

```
sudo lpadmin -x имя_принтера
```

6.7.1.1 Настройка сетевого принтера (IPP)

Настройка сетевого принтера с использованием IPP требует выполнения ряда шагов через веб-интерфейс CUPS, а также некоторых изменений через командную строку, если это необходимо. Для настройки сетевого принтера необходимо:

а) Открыть веб-интерфейс CUPS по адресу http://localhost:631;

б) Перейти на вкладку "Администрирование" и выбрать опцию "Добавить принтер". Для доступа к разделу "Администрирование" необходимо ввести учетные данные администратора для подтверждения действий;

в) В списке доступных принтеров нужно выбрать "Internet Printing Protocol (IPP)" или "IPP Everywhere" (в зависимости от модели принтера);

г) Ввод данных о принтере:

– в поле "Подключение" следует ввести IP-адрес или URL принтера;

– Система попытается определить доступные параметры принтера автоматически. Также необходимо убедиться, что настройки IPP верны.

Выбор драйвера и настройка параметров:

– в процессе настройки CUPS предложит выбрать подходящий драйвер. В большинстве случаев он будет выбран автоматически. Если целевой принтер не был найден автоматически, необходимо выбрать драйвер из списка вручную.

– при необходимости можно настроить дополнительные параметры: формат бумаги, разрешение, цветность и другие.

Завершение настройки:

– после добавления принтера перейти в раздел "Принтеры" для проверки статуса устройства;

– печать пробной страницы для проверки правильности конфигурации.

Некоторые параметры также можно настроить напрямую через командную строку с использованием команды `lpadmin`:

```
sudo lpadmin -p office-printer -E -v ipp://192.168.1.100/printers/office-printer -m everywhere
```

где:

– -p – указывает имя принтера,
– -v – URL-адрес подключения,
– -m – драйвер.

Для включения принтера и активации его в Системе нужно выполнить команды:

```
sudo cupsenable office-printer
sudo cupsaccept office-printer
```

Для просмотра состояния задания печати и устранения неполадок можно использовать как веб-интерфейс, так и команды:

Просмотр очереди заданий:

```
lpstat -o
```

Отмена задания:

```
cancel job-id
```

Для диагностики проблем с печатью нужно проверить логи CUPS, расположенные в /var/log/cups/. В случае возникновения проблем с доступом к принтеру или драйверами, нужно убедиться, что все необходимые пакеты установлены и службы работают корректно.

```
systemctl status cups
```

Если возникают проблемы с подключением или печатью, рекомендуется проверить логи CUPS, которые хранятся в /var/log/cups/error_log.

6.7.1.2 Управление доступом к принтеру

CUPS позволяет настраивать доступ к принтерам для отдельных пользователей или групп. Это можно сделать как через веб-интерфейс, так и в конфигурационных файлах.

Управление доступом и правами через веб-интерфейс:

а) В разделе "Принтеры" необходимо выбрать нужный принтер;

б) Затем перейти в меню "Настроить разрешенных пользователей" для ограничения доступа к принтеру;

в) Здесь можно указать конкретных пользователей или группы, которые имеют право на печать.

Чтобы ограничить доступ к принтеру через консоль, нужно добавить в конфигурацию следующие строки:

```
<Location /printers/имя_принтера>
Order deny,allow
Deny from all
Allow from 192.168.1.0/24
</Location>
```

где параметр Order deny,allow задает порядок обработки правил. Сначала применяются правила deny, а затем правила allow. Это значит, что сначала доступ запрещается для всех, а затем разрешается только для указанных источников.

После внесения изменений нужно сохранить их и перезапустить CUPS:

```
sudo systemctl restart cups
```

6.8 Почтовый сервер OpenSMTPD

OpenSMTPD представляет собой почтовый сервер, развиваемый проектом OpenBSD и предназначенный для простой, безопасной и надежной замены таких

решений, как Sendmail. Данный сервер обеспечивает минимальную конфигурацию при сохранении всех необходимых функций.

Для установки OpenSMTPD необходимо выполнить следующую команду:

```
sudo dnf install opensmtpd
```

Рекомендуется ознакомиться с составом пакета, включая список доступных руководств (man-страниц):

```
rpm -ql opensmtpd
```

Запуск службы осуществляется командой:

```
sudo systemctl start opensmtpd
```

Для проверки состояния службы используется:

```
sudo systemctl status opensmtpd
```

Журнал работы службы просматривается командой:

```
sudo journalctl -u opensmtpd
```

Для проверки используемых интерфейсов и портов следует выполнить:

```
sudo ss -ntulp | grep smtpd
```

По умолчанию служба прослушивает порт 25 на локальном интерфейсе (127.0.0.1).

При успешном запуске службы необходимо добавить её в автозапуск:

```
sudo systemctl enable opensmtpd
```

6.8.1 Настройка OpenSMTPD

Для корректной работы почтового сервера следует задать его доменное имя, например:

```
sudo hostnamectl set-hostname mail.rosa.loc
```

В приведённом примере в качестве имени хоста используется mail.rosa.loc, что подразумевает обслуживание почтовых адресов вида user@rosa.loc (где rosa.loc — домен второго уровня).

Файл конфигурации OpenSMTPD расположен по адресу /etc/opensmtpd/smtpd.conf.

Для редактирования файла конфигурации нужно выполнить:

```
sudo nano /etc/opensmtpd/smtpd.conf
```

По умолчанию в конфигурации задано значение, которое определяет, что служба smtpd будет слушать только соединения, приходящие с локального компьютера:

```
listen on localhost
```

Данный параметр регламентирует, что:

– входящие подключения из внешней сети будут недоступны;

– почта может отправляться и приниматься только локально, например, между системными службами или пользователями на этом же сервере.

Если предполагается доступность почтового сервера из внешней сети, следует заменить эту опцию на следующую:

```
listen on 0.0.0.0
```

Для указания домена для обслуживания в файле конфигурации необходимо над строкой:

```
match for local action "local"
```

добавить строку, определяющую домен:

```
match from any for domain "rosa.loc" action "local"
```

где rosa.loc — это доменное имя, соответствующее ранее заданному имени хоста.

После внесения изменений в конфигурацию необходимо:

а) перезапустить службу командой:

```
sudo systemctl restart opensmtpd
```

б) убедиться в её успешном запуске:

```
sudo systemctl status opensmtpd
```

6.8.2 Проверка работы OpenSMTPD

После настройки конфигурации почтового сервера необходимо убедиться в его работоспособности. По умолчанию OpenSMTPD осуществляет доставку писем в формате Maildir, где каждое сообщение хранится в отдельном файле в каталоге ~/Maildir домашней директории пользователя-получателя. Этот каталог требуется создать заранее.

Для осуществления проверки доставки почты выполняют следующие шаги:

а) Создание нового пользователя-получателя:

```
sudo useradd ivanov
```

б) Создание каталога для хранения почты:

```
sudo -u ivanov mkdir -p /home/ivanov/Maildir
```

в) Определение IP-адреса сервера командой:

```
ip a
```

Например, адрес сервера: 192.168.122.12.

г) Проверка доставки письма с другого ПК на сервер через службу Telnet:

```
telnet 192.168.122.12 25
```

Ожидаемый ответ сервера:

```
220 mail.rosa.loc ESMTP OpenSMTPD
```

д) Последовательное выполнение команд SMTP-протокола (с обязательным соблюдением синтаксиса):

```
helo dom.loc
```

Ожидаемый ответ:

```
250 mail.rosa.loc Hello dom.loc [192.168.122.1], pleased to meet you
```

Следующей командой указывается адрес отправителя письма (следует обратить внимание на необходимость строгого соблюдения RFC 2822 и указания адреса в знаках < >):

```
mail from:<vasya@domain.loc>
```

Ожидаемый ответ:

```
250 2.0.0 Ok
```

Указывается получатель письма:

```
rcpt to:<ivanov@rosa.loc>
```

Ожидаемый ответ:

```
250 2.1.5 Destination address valid: Recipient ok
```

Ввод:

```
data
```

В ответ приходит приглашение начать ввода письма. Необходимо ввести текст:

```
Message-ID: <x01@dom.loc>
Subject: Test

Privet
```

```
.
```

Ожидаемый ответ:

```
250 2.0.0 <message_id> Message accepted for delivery
```

Проверка логов и доставки сообщения:

1) Просмотр системного журнала командой:

```
journalctl -u opensmtpd
```

2) Далее нужно убедиться, что письмо сохранено в директории пользователя:

```
cat /home/ivanov/Maildir/new/*
```

Пример содержимого доставленного сообщения:

```
Return-Path: <vasya@domain.loc>
Delivered-To: ivanov@rosa.loc
Received: from dom.loc (hp-xfce [192.168.122.1])
     by mail.rosa.loc (OpenSMTPD) with SMTP id df9164e0
     for <ivanov@rosa.loc>;
     Sat, 10 Dec 2022 20:13:31 +0000 (UTC)
Message-ID: <x01@dom.loc>
Subject: Test

Privet
```

Почтовый сервер определяет домашний каталог пользователя через механизм PAM. Конфигурация PAM для службы OpenSMTPD расположена в файле:

```
/etc/pam.d/opensmtpd
```

Домашний каталог пользователя ivanov был получен через PAM. Конфигурация PAM для службы OpenSMTPD расположена в файле:

```
/etc/pam.d/opensmtpd
```

Это позволяет использовать как локальные, так и доменные (например, LDAP) учетные записи для получения почты.

6.8.3 Интеграция OpenSMTPD с Dovecot

Для организации полноценного почтового сервера, обеспечивающего как приём сообщений (SMTP), так и их выдачу пользователям (IMAP), целесообразно использовать связку OpenSMTPD и Dovecot. В данной конфигурации OpenSMTPD будет принимать почту и сохранять её в формате Maildir, а Dovecot — предоставлять доступ к этой почте по IMAP.

Для установки Dovecot с необходимыми зависимостями следует выполнить команду:

```
sudo dnf install dovecot
```

Для настройки Dovecot необходимо выполнить следующие шаги:

а) Открыть и перейти в основной конфигурационный файл:

```
sudo nano /etc/dovecot/conf.d/10-mail.conf
```

б) В файле необходимо найти строку:

```
#mail_location =
```

Далее нужно заменить найденную строку на следующую:

```
mail_location = maildir:~/Maildir
```

Внесенные опции укажут Dovecot на использование формата хранения Maildir в домашнем каталоге каждого пользователя.

в) После внесения изменений необходимо запустить и активировать службу Dovecot командой:

```
sudo systemctl enable --now dovecot.socket
```

После выполненных настроек почта будет работать следующим образом:

– OpenSMTPD принимает сообщения по протоколу SMTP и сохраняет их в директорию ~/Maildir пользователя;

– Dovecot предоставляет пользователю доступ к этим сообщениям по протоколу IMAP.

Таким образом можно подключаться к почтовому ящику как по SMTP (для исходящей почты), так и по IMAP (для входящей почты), например, используя удобный почтовый клиент.

По умолчанию как OpenSMTPD, так и Dovecot используют системных пользователей Linux и их пароли для аутентификации. Это обеспечивает интеграцию с PAM и возможность управления учетными записями через стандартные механизмы ОС.

6.8.4 Дополнительные рекомендации по безопасности и интеграции

Для обеспечения безопасной работы почтового сервера OpenSMTP также рекомендуется выполнять следующие настройки Системы:

а) Настройка TLS/SSL. Для обеспечения защищённой передачи писем рекомендуется настроить TLS/SSL на OpenSMTPD и Dovecot.

В smtpd.conf можно активировать STARTTLS или слушать SMTPS (порт 465), задав путь к сертификатам и ключам:

```
listen on 0.0.0.0 port 587 tls pki mail.rosa.loc
```

где pki mail.rosa.loc — определение сертификата в конфигурации OpenSMTPD.

Можно сгенерировать самоподписанные сертификаты либо использовать сервис Let's Encrypt (центр сертификации, предоставляющий бесплатные криптографические сертификаты X.509 для шифрования передаваемых через интернет данных HTTPS и других протоколов, используемых серверами в Интернете).

Для IMAP рекомендуется настроить Dovecot на порт 993 (IMAPS), в файле /etc/dovecot/conf.d/10-ssl.conf, указав:

```
ssl = required
ssl_cert =
</etc/letsencrypt/live/mail.rosa.loc/fullchain.pem
ssl_key = </etc/letsencrypt/live/mail.rosa.loc/privkey.pem
```

В результате пароли пользователей будут передаваться по защищённому каналу.

Для защиты пользователей почты рекомендуется интеграция со спам-фильтрами и антивирусными решениями. В качестве антиспам-решения может быть установлен rspamd или SpamAssassin. Для проверки вложений на наличие вирусов можно интегрировать ClamAV.

В цепочке обработки писем OpenSMTPD настраивается фильтр с помощью filter-директив, перенаправляющий почту через rspamd или amavis. Эти компоненты могут быть развернуты на том же сервере или вынесены на отдельные узлы.

Настройка SPF, DKIM и DMARC:

– SPF настраивается через публикацию DNS-записи типа TXT, в которой перечисляются IP-адреса отправителей, например:

```
v=spf1 ip4:192.168.122.12 -all
```

– DKIM предполагает установку пакета opendkim и генерацию пары ключей, после чего публичный ключ публикуется в DNS;

– DMARC задаётся также через TXT-запись в DNS и позволяет указать, как обрабатывать письма, не прошедшие SPF/DKIM.

Эти механизмы совместно снижают риск попадания писем в папки спама у получателей.

Настройка ротации логов. Журналы работы OpenSMTPD и Dovecot могут со временем занимать значительный объём, для их ротации используется система logrotate. В стандартных пакетах уже предусмотрены конфигурации в каталоге /etc/logrotate.d/. При необходимости можно настроить собственные параметры, например: частоту ротации, количество архивных копий и алгоритм сжатия. Регулярная ротация позволяет не переполнить файловую систему логами.

Рекомендуется регулярно выполнять резервное копирование:

– конфигурационных файлов (/etc/opensmtpd/, /etc/dovecot/);
– почтовых ящиков пользователей (~/Maildir);
– базы пользователей (если используется внешняя БД).

Для автоматизации можно применять rsync, borg, tar в связке с cron.

Настройка межсетевого экранирования (Nftables) для почтовых сервисов.

Для обеспечения доступности SMTP и IMAP/IMAPS из внешней сети необходимо настроить правила в системе сетевого экранирования Nftables, разрешив входящие соединения на соответствующие порты:

– 25 – SMTP (OpenSMTPD);
– 143 – IMAP (Dovecot);
– 993 – IMAPS (если используется шифрование).

Пример добавления правил для OpenSMTP в Nftables:

а) Открыть файл конфигурации Nftables:

```
sudo nano /etc/nftables.conf
```

б) Добавить правила в таблицу inet (если такая уже создана) или в существующую filter:

```
table inet filter {
    chain input {
        type filter hook input priority 0;

        # Разрешить входящие подключения к почтовым сервисам
        tcp dport {25, 143, 993} accept
    }
}
```

в) Применить изменения:

```
sudo nft flush ruleset
sudo nft -f /etc/nftables.conf
```

г) Проверить применённые правила:

```
sudo nft list ruleset
```

Также рекомендуется выполнить следующие меры для обеспечения безопасности подключений почтового сервера:

– ограничить доступ к портам 25/143/993 только с доверенных IP-адресов (например, только адреса организации или VPN);

– при необходимости использовать шифрование для SMTP (порт 465 или STARTTLS на 25) и IMAP (порт 993);

– убедиться, что правила Nftables применяются при каждом запуске Системы командой:

```
systemctl enable nftables
```

6.9 Почтовый сервер Postfix

Postfix является почтовым транспортным агентом (MTA), предназначенным для приёма, обработки и доставки электронных сообщений. Система обеспечивает маршрутизацию входящей и исходящей почты, взаимодействуя как с локальными пользовательскими процессами, так и с удалёнными почтовыми серверами по протоколу SMTP.

Postfix построен по модульной архитектуре и включает набор специализированных служб, выполняющих отдельные функции: обработку входящих соединений (smtpd), доставку сообщений (smtp), управление очередями сообщений (pickup, qmgr) и вспомогательные операции. Работу всех служб координирует основной процесс master.

Очередь сообщений является центральным элементом обработки почты. В неё помещаются входящие и исходящие сообщения до момента их доставки. Если доставка невозможна в текущий момент (например, сервер получателя недоступен), сообщение остаётся в очереди до повторной попытки.

6.9.1 Установка и начальная подготовка

Пакет Postfix устанавливается из официального репозитория ОС командой:

```
sudo dnf install postfix
```

В некоторых случая может понадобиться использование аргумента –alloweresing с помощью которого предотвращается конфликт установленных пакетов в Системе.

После установки необходимо добавить службу Postfix к автозагрузке в systemd:

```
systemctl enable postfix
```

Для запуска сервиса используется команда:

```
systemctl start postfix
```

Для проверки статуса службы необходимо выполнить команду:

```
sudo systemctl status postfix
```

Вывод на экран строки "Active: active (running)" сообщает о том, что Postfix успешно запущен и работает.

Перед началом конфигурации необходимо проверить корректность сетевого имени узла. Полное доменное имя (FQDN) должно быть определено Системой и соответствовать DNS-записям. Для проверки параметра используются команды:

```
hostname -f
hostname -d
```

В случае необходимости имя узла настраивается через команду `hostnamectl`, а соответствующие записи добавляются в /etc/hostname и /etc/hosts. Для рабочего почтового сервера требуется корректное разрешение FQDN, включая наличие A-записи, MX-записи и обратной PTR-записи.

6.9.2 Основные параметры конфигурации

Базовая настройка Postfix выполняется в файле конфигурации /etc/postfix/main.cf, в котором для удобства чтения указываются и вносятся только параметры, изменённые администратором, а также те значения по умолчанию, которые требуется корректировать в процессе настройки.

Полный перечень стандартных параметров приведён в справочном файле main.cf.default, не подлежащем редактированию.

Дополнительная расширенная информация с комментариями содержится в файле main.cf.dist. Наиболее полное описание всех параметров конфигурации доступно в встроенной справке команды `postconf`:

```
postconf -h
```

После внесения изменений в конфигурационный файл необходимо перезапустить службу Postfix для применения новых настроек:

```
sudo systemctl restart postfix
```

Основные параметры настройки файла main.cf:

– myhostname — полное доменное имя почтового сервера (FQDN), например:

```
myhostname = mail.example.org.
```

– mydomain — доменная часть имени хоста, например:

```
mydomain = example.org.
```

– myorigin — домен, подставляемый к локальной почте при отправке, обычно формируется из имени домена указанного в параметре mydomain; используется для того, чтобы локальный пользователь user1, отправляющий письмо без указания домена, был идентифицирован как user1@$mydomain. Параметр задается в формате:

```
myorigin = $mydomain.
```

– mydestination — перечень доменов, для которых сервер выполняет локальную доставку, например:

```
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain.
```

– inet_interfaces — сетевые интерфейсы, на которых Postfix принимает входящие SMTP-соединения. По умолчанию задан параметр inet_interfaces = localhost, при котором Postfix работает в локальном режиме и не принимает входящие SMTP-соединения с внешних узлов. Для включения полноценного режима SMTP-сервера необходимо разрешить обработку сообщений со всех узлов:

```
inet_interfaces = all
```

– inet_protocols — поддерживаемые сетевые протоколы (IPv4/IPv6), например:

```
inet_protocols = ipv4.
```

– mynetworks — список доверенных сетей, из которых разрешена пересылка почты без авторизации. Рекомендуется указать весь диапазон адресов, относящихся только к локальному компьютеру:

```
mynetworks = 127.0.0.0/8
```

– relayhost — SMTP-сервер, используемый для передачи исходящей почты в случае работы через почтовый шлюз или провайдера. По умолчанию параметр не используется и добавляется только при необходимости.

После изменения конфигурации требуется выполнить проверку конфигурационного файла:

```
postfix check
```

Вывод команды укажет на ошибки в работе сервиса, если они есть.

Для применения изменений конфигурации необходимо перезапустить службу Postfix:

```
sudo systemctl restart postfix
```

6.9.3 Настройка исходящей почты (SMTP-клиент)

Для отправки почты внешним адресатам Postfix может работать как самостоятельный SMTP-сервер или использовать внешний почтовый шлюз (relay-сервер) провайдера или корпоративной системы.

Внешний почтовый шлюз задается параметром relayhost. Пример использования SMTP-сервера провайдера:

```
relayhost = [smtp.example.org]:587
```

где 587— это стандартный порт отправки почты клиентом на SMTP-сервер с обязательной аутентификацией пользователя.

Если внешний сервер требует аутентификации, необходимо включить механизм SASL:

```
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_use_tls = yes
smtp_tls_security_level = encrypt
```

Файл /etc/postfix/sasl_passwd должен содержать учетные данные в формате:

```
[smtp.example.org]:587 user@example.org:Пароль
```

После заполнения файла выполняется его подготовка:

```
postmap /etc/postfix/sasl_passwd
chmod 600 /etc/postfix/sasl_passwd*
```

Если внешний SMTP-сервер не используется, Postfix осуществляет доставку напрямую. В этом случае требуется корректная DNS-конфигурация домена, включая записи MX и PTR. Для проверки DNS-конфигураций используются следующие команды:

– для проверки MX-записи:

```
dig +short MX example.org
```

Ожидаемый вывод формата:

```
10 mail.example.org
```

– для проверки A-записи:

```
dig +short mail.example.org A
```

В выводе отображается IP-адрес, который соответствует A-записи указанного домена:

```
192.0.2.15
```

– для проверки PTR-записи:

```
dig +short -x 192.0.2.15
```

Ожидаемый вывод в формате:

```
mail.example.org
```

A-запись и PTR-запись должны указывать друг на друга.

После изменения параметров необходимо выполнить проверку конфигурации и перезапуск службы:

```
postfix check
systemctl restart postfix
```

6.9.4 Настройка входящей почты

Для того чтобы Postfix мог принимать входящие SMTP-сообщения с других серверов в качестве полноценного почтового сервера, необходимо активировать прослушивание всех сетевых интерфейсов и определить обслуживаемые им домены.

Для этого в основном конфигурационном файле /etc/postfix/main.cf необходимо указать параметр:

```
inet_interfaces = all
```

Определение обслуживаемых локальных доменов задается в параметре mydestination:

```
mydestination = $myhostname, $mydomain, localhost.$mydomain, localhost
```

Этот параметр определяет Postfix конечным пунктом доставки для заданных доменов (т. е. приемом почты в локальные ящики), а не ретранслятором почты дальше.

Для корректной работы входящей почты крайне необходимо наличие правильных DNS-конфигураций (A-записи, указывающей IP-адрес вашего сервера, и соответствующей MX-записи для домена, направляющей почту на этот сервер). Проверка DNS-конфигураций рассмотрена в п. 6.9.3.

Для приёма входящих SMTP-соединений Postfix должен быть доступен по TCP-порту 25. Для этого производится настройка nftables, которая разрешает входящие соединения на порт 25/TCP:

```
nft add rule inet filter input tcp dport 25 accept
```

Это правило следует внести в постоянную конфигурацию nftables (в конфигурационный файл /etc/nftables.conf), чтобы обеспечить сохранение настроек после перезагрузки. Подробнее о настройках nftables см. раздел 5.6.

После настройки необходимо выполнить проверку конфигурации и перезапустить службу с правами администратора:

```
postfix check
systemctl restart postfix
```

6.9.5 Настройка безопасности

Для обеспечения безопасной работы почтового сервера необходимо настроить шифрование соединений, ограничения на приём входящих SMTP-запросов и механизм аутентификации клиентов.

Postfix поддерживает использование шифрования соединений TLS для защиты передаваемых данных. Для включения шифрования необходимо указать пути к сертификату и закрытому ключу:

```
smtpd_tls_cert_file = /etc/ssl/certs/postfix.crt
smtpd_tls_key_file = /etc/ssl/private/postfix.key
smtpd_tls_security_level = may
smtpd_tls_loglevel = 1
```

Уровень may включает TLS при наличии поддержки со стороны клиента. При необходимости можно использовать уровень encrypt, который требует обязательного шифрования соединения.

Для исходящих SMTP-соединений рекомендуется использовать шифрование:

```
smtp_tls_security_level = encrypt
smtp_tls_loglevel = 1
```

Для предотвращения несанкционированной пересылки сообщений необходимо задать ограничения для входящих SMTP-запросов:

```
smtpd_helo_required = yes
smtpd_sender_restrictions = reject_unknown_sender_domain
smtpd_recipient_restrictions =
        permit_mynetworks,
        reject_unauth_destination
```

Данные параметры обеспечивают проверку домена отправителя и запрещают пересылку почты в домены, которые не обслуживаются сервером.

Для защиты от некорректных SMTP-диалогов рекомендуется использовать минимальные ограничения на команду HELO:

```
smtpd_helo_restrictions = reject_invalid_helo_hostname
```

Для разрешения пересылки почты пользователями, находящимися вне сети сервера, включается механизм SASL-аутентификации:

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
```

При использовании внешнего SMTP-сервера (например, корпоративного сервера) параметры SASL на стороне клиента указываются в разделе исходящей почты.

При использовании таблиц nftables в качестве сетевого экранирования необходимо убедиться, что правила фильтрации разрешают TLS-подключения (TCP-порт 587 или 465, если используется SMTPS):

– пример для порта 587:

```
nft add rule inet filter input tcp dport 587 accept
```

– если используется SMTPS (порт 465):

```
nft add rule inet filter input tcp dport 465 accept
```

Правила следует добавить в постоянную конфигурацию /etc/nftables.conf.

После настройки параметров безопасности необходимо выполнить проверку конфигурации и перезапуск Postfix:

```
postfix check
systemctl restart postfix
```

6.9.6 Управление очередями сообщений

Postfix использует несколько очередей для хранения сообщений на различных этапах обработки. Очереди содержат письма, ожидающие доставки, повторной попытки отправки или удаления. Управление очередями осуществляется утилитой postqueue (входит в пакет Postfix) и набором диагностических команд.

Для просмотра состояния очереди используется команда:

```
postqueue -p
```

Удаление всех сообщений из очереди:

```
postsuper -d ALL
```

Удаление только "зависших" или недоставляемых сообщений:

```
postsuper -d ALL deferred
```

Принудительная попытка доставки всех сообщений в очереди:

```
postqueue -f
```

Информация о состоянии очереди и попытках доставки отражается в системном журнале Postfix:

```
journalctl -u postfix
```

6.9.7 Проверка работы и диагностика

После завершения настройки рекомендуется проводить проверку исходящей и входящей почты, а также анализ журналов службы Postfix.

Для первичной проверки корректности конфигурационных файлов используется команда:

```
postfix check
```

При наличии ошибок утилита выводит сведения о неисправных параметрах или отсутствующих файлах.

После внесения изменений в конфигурацию необходимо перезапустить службу Postfix:

```
systemctl restart postfix
```

Статус работы службы просматривается командой:

```
systemctl status postfix
```

Для тестовой отправки сообщения используется утилита sendmail, входящая в состав Postfix:

```
echo "Тестовое сообщение" | sendmail -v user@example.org
```

Вывод команды позволяет оценить корректность SMTP-диалога и успешность доставки.

Проверка входящих соединений производиться с удалённого хоста:

```
telnet <IP-сервера> 25
```

При корректной работе Postfix возвращает SMTP-ответ вида:

```
220 mail.example.org ESMTP Postfix
```

Журналы Postfix доступны через systemd-журнал:

```
journalctl -u postfix
```

Для просмотра последних событий Postfix:

```
journalctl -u postfix -n 50
```

Наличие ошибок доставки, отказов соединения или проблем с TLS можно определить по сообщениям в журнале.

6.9.8 Примеры типовых конфигураций

Далее в пп. 6.9.8.1–6.9.8.5 рассмотрены примеры типовых конфигураций файла main.cf в зависимости от пользовательских потребностей.

6.9.8.1 Локальный режим

Локальный режим используется на рабочих станциях и серверах, не предназначенных для приёма внешней почты:

```
myhostname = host.example.org
mydomain = example.org
myorigin = $mydomain
inet_interfaces = localhost
mynetworks = 127.0.0.0/8
mydestination = localhost
```

При таких настройках конфигурации Postfix принимает сообщения только от локальных процессов и выполняет исходящую доставку через прямые подключения или внешний SMTP-сервер, если задан параметр relayhost.

6.9.8.2 Использование внешнего SMTP-сервера для отправки почты

Данная конфигурация может быть использована для организации почтового агента, при которой исходящая почта должна отправляться через корпоративный или провайдерский сервер:

```
relayhost = [smtp.provider.org]:587
smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_use_tls = yes
smtp_tls_security_level = encrypt
```

Формат файла /etc/postfix/sasl_passwd:

```
[smtp.provider.org]:587 user@example.org:Пароль
```

6.9.8.3 Полноценный почтовый сервер с приёмом входящей почты

Параметры конфигурации в случаях, когда Postfix обслуживает домен и принимает сообщения извне:

```
    myhostname = mail.example.org
    mydomain = example.org
    myorigin = $mydomain
    inet_interfaces = all
    mydestination = $myhostname, $mydomain, localhost.$mydomain,
localhost
    mynetworks = 127.0.0.0/8
    smtpd_helo_required = yes
    smtpd_sender_restrictions = reject_unknown_sender_domain
    smtpd_recipient_restrictions =
           permit_mynetworks,
           reject_unauth_destination
```

Для корректной работы требуется наличие MX-, A- и PTR-записей в DNS.

6.9.8.4 Включение TLS для входящих и исходящих соединений

Параметры конфигурации для настройки TLS-шифрования:

```
    smtpd_tls_cert_file = /etc/ssl/certs/postfix.crt
    smtpd_tls_key_file = /etc/ssl/private/postfix.key
    smtpd_tls_security_level = may
    smtp_tls_security_level = encrypt
```

Параметр encrypt обеспечивает обязательное шифрование при исходящих SMTP-сеансах.

6.9.8.5 Минимальная конфигурация для тестового стенда

Параметры конфигурации для локальной проверочной среды, не предназначенной для эксплуатации:

```
    myhostname = test.local
    inet_interfaces = localhost
    mydestination = localhost
    mynetworks = 127.0.0.0/8
```

7 БЕЗОПАСНОСТЬ

7.1 Контроль доступа к файлам

7.1.1 Общие сведения о правах доступа

Контроль доступа к объектам файловой системы является одним из ключевых механизмов защиты данных в ОС. Каждому файлу и каталогу в Системе назначаются права, определяющие, какие действия могут выполнять различные категории пользователей.

Права доступа к файлам и каталогам в ОС разделяются на три категории:

– Владелец (user): права доступа для пользователя, которому принадлежит файл.

– Группа (group): права доступа для группы пользователей, связанной с файлом.

– Остальные (others): права для всех остальных пользователей Системы.

Права доступа включают в себя:

– r (read) – чтение: пользователь может просматривать содержимое файла или список файлов в каталоге.

– w (write) – запись: пользователь может изменять содержимое файла или создавать, удалять и переименовывать файлы в каталоге.

– x (execute) – выполнение: пользователь может запускать файл на выполнение (если это исполняемый файл или скрипт) или переходить в каталог.

Права доступа также можно задать с помощью чисел. Каждому праву доступа соответствует определенное число:

– r = 4;
– w = 2;
– x = 1.

Суммируя эти значения, можно задать конкретные права в числовом формате. Например:

– 7 (rwx) = 4 (чтение) + 2 (запись) + 1 (выполнение);
– 5 (r-x) = 4 (чтение) + 1 (выполнение).

Общая схема настройки прав для пользователей и групп пользователей одинакова и для файлов, и для папок.

Система отображает права доступа в выводе команды `ls -l`, которая показывает содержимое директории (рисунок 110):

```
rosa-fk7edm user # ls -l /etc
итого 3000
-rw-r--r--   1 root root        16 апр 11 14:56 adjtime
-rw-r--r--.  1 root root       732 дек  1  2022 aliases
-rw-r-----.  1 root mail     12288 мар 29 02:37 aliases.db
drwxr-xr-x.  3 root root      4096 мар 29 02:34 alsa
drwxr-xr-x.  2 root root      4096 мар 29 02:38 alternatives
-rw-r--r--.  1 root root       559 дек 26 14:28 anacrontab
drwxr-xr-x.  2 root root      4096 мар 29 02:34 apparmor.d
-rw-r-----.  1 root daemon       1 сен 30  2021 at.deny
drwxr-xr-x.  3 root root      4096 мар 29 02:35 autofs
drwxr-xr-x.  3 root root      4096 мар 29 02:35 avahi
drwxr-xr-x.  2 root root      4096 мар 29 02:39 bash_completion.d
-rw-r--r--.  1 root root      2218 янв 22 03:57 bashrc
-rw-r--r--.  1 root root       535 мая 10  2021 bindresvport.blacklist
drwxr-xr-x.  2 root root      4096 дек  5 04:41 binfmt.d
drwxr-xr-x.  2 root root      4096 окт 29  2022 blkid
drwxr-xr-x.  2 root root      4096 мар 29 02:35 bluetooth
drwxr-xr-x.  2 root root      4096 мар 29 02:33 branding-configs
drwxr-xr-x.  2 root root      4096 мар 27 23:17 chromium-browser
-rw-r--r--.  1 root root      1362 мая 12  2021 chrony.conf
-rw-r-----.  1 root chrony     540 мая 12  2021 chrony.keys
drwxr-xr-x.  2 root root      4096 мар 29 02:35 cifs-utils
drwxr-xr-x.  2 root root      4096 дек 26 14:28 cron.d
```

Рисунок 110 – Вывод команды `ls -l`

Первые 10 символов в строке содержат информацию о правах к файлу или каталогу. Далее следует имя пользователя-владельца и группы-владельца.

Пример вывода команды `ls -l`:

```
-rw-r--r--
```

Где параметры:

– rw- – владелец может читать и изменять файл;
– r-- – группа может только читать файл;
– r-- – остальные пользователи также могут только читать файл.

Назначение владельца и группы отображается при выполнении команды `ls -l`:

```
-rw-r----- 1 root adm 1024 May 10 12:00 syslog
```

где:

– root — владелец,
– adm — группа.

7.1.2 Смена владельца файла или каталога

Для смены владельца файла или каталога доступны два инструмента. Первый – `chown`, позволяет изменить пользователя и группу файла или папки:

```
chown [новый пользователь]:[новая группа] [файл или папка]
```

Если не указывать группу, изменится только пользователь-владелец. Если не указывать пользователя (:[новая группа]), изменится только группа-владелец.

При изменении прав на директории можно использовать параметр -R. Он рекурсивно изменит владельца всех вложенных директорий и файлов. Если представить, что нужно предоставить права на папку new-user для одноимённого пользователя и его группы, то команда для этого будет выглядеть так:

```
chown -R new-user:new-group /home/users/new-user
```

Второй инструмент – `chgrp`. В отличие от первого, меняет только группу-владельца:

```
chgrp [группа] [файл или папка]
```

Список распространённых параметров команды `chgrp`:

– -h – работа непосредственно с самими символьными ссылками;
– --dereference – работа с файлами, а не самими символьными ссылками (используется по умолчанию);
– -R – рекурсивная обработка каталога со всем его содержимым;
– -H – переход по символической ссылке и изменение атрибутов файла/каталога; используется вместе с параметром -R;
– -L – переход по символической ссылке и продолжение рекурсивной обработки; используется вместе с параметром -R;
– -P – обработка только символической ссылки при наличии; используется вместе с параметром "-R"; является значением по умолчанию;
– --reference=имя_образца – использование группы образца; используется вместо группы;
– -c – вывод при обработке только изменений;
– -v – вывод информации о каждом обработанном объекте.

7.1.3 Изменение прав доступа

Для работы непосредственно с правами используется команда `chmod`:

```
chmod [настройки прав] [файл или папка]
```

При работе с директориями опция -R позволит изменить права на все вложенные файлы и папки.

Настройки прав в chmod можно определять двумя способами:

а) указать категорию (u – пользователь-владелец, g – группа-владелец, o – другие пользователи, a – все пользователи), модификатор ("+", " ", "=") и, соответственно, нужные права (r, w, x).

Например, если есть файл example.txt с максимальным уровнем прав для всех категорий пользователей rwx rwx rwx. Если не нужно, чтобы кто-то мог запускать этот файл на выполнение, нужно убрать параметр x из прав всех категорий пользователей сразу. Это можно сделать так:

```
chmod a-x example.txt
```

Права доступа к файлу поменяются на:

```
rw-rw-rw-
```

Теперь, если необходимо вернуть владельцу права на запуск файла, следует добавить параметр x в категорию пользователя-владельца:

```
chmod u+x example.txt
```

После чего права доступа на файл станут следующими:

```
rwxrw-rw-
```

Для изменения всего набора параметров для отдельной категории сразу нужно использовать команду:

```
chmod u=rwx example.txt
```

Права доступа к файлу снова изменятся на:

```
rwxrwxrwx
```

б) указать права можно в виде числового значения, что является более сложным, но более быстрым способом. В этом случае в качестве параметра нужно передать цифровое выражение уровня прав:

```
chmod 777 example.txt
```

В этом примере назначен максимальный уровень доступа всем категориям пользователей.

Соответственно, для изменения уровня прав отдельной категории пользователей нужно изменить только это числовое значение. Например, чтобы запретить исполнение файла для всех пользователей:

```
chmod 666 example.txt
```

Чтобы вернуть права, но только пользователю-владельцу:

```
chmod 766 example.txt
```

Если необходимо изменить права только на все папки или только на все файлы в определённой директории, например, при настройке прав на файлы сайтов, можно использовать один из вариантов:

– Смена прав на 755 для всех папок внутри /var/www/sites/example.com:

```
find /var/www/sites/example.com/ -type d -exec chmod 755 {} \;
```

– Смена прав на 644 для всех файлов внутри /var/www/sites/example.com:

```
find /var/www/sites/example.com/ -type f -exec chmod 644 {} \;
```

7.1.4 Изменение атрибутов файлов

Помимо прав доступа и владельца, каждый файл может иметь ряд атрибутов, определяемых на уровне файловой системы. Атрибуты показывают, какие операции могут или не могут проводиться с файлом независимо от владельца.

Атрибуты файлов в текущей директории можно просмотреть с помощью команды `lsattr`. Команда без аргументов выведет атрибуты всех файлов в текущей директории. Если указать путь к файлу или папке, будут перечислены свойства указанного файла или списка файлов в указанной папке соответственно (рисунок 111):

```
lsattr example.txt
```

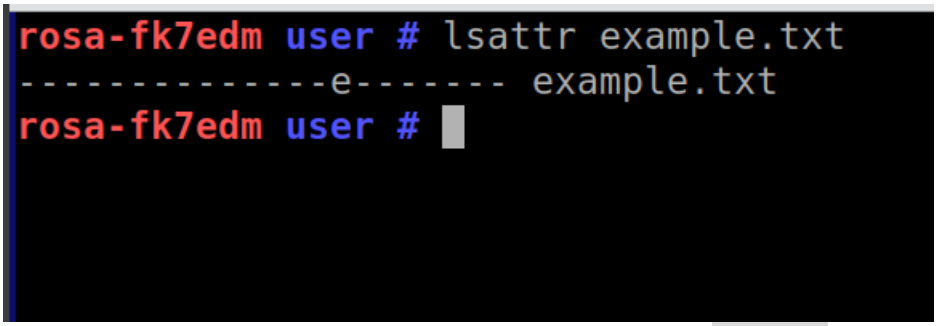


Рисунок 111 - Использование команды `lsattr`

Первые 20 символов в строке предназначены для отображения атрибутов файла.

Список атрибутов может отличаться в зависимости от файловой системы:

– i – файл становится неуязвим для любых изменений ("godmode"). Его нельзя удалить, переименовать, изменить содержимое или создать на него символическую ссылку.

– a – в файл можно только добавлять новые данные. Изменение или удаление существующего содержимого запрещено. Атрибут часто используется для защиты лог-файлов. Изменять такие файлы может только суперпользователь root.

– s – при удалении файл безвозвратно стирается: все блоки на диске, которые он занимал, перезаписываются нулями.

– u – активирует возможность восстановления содержимого файла после удаления ("обратимое" удаление).

– c – файл хранится в сжатом виде: при записи данные автоматически сжимаются, при чтении – распаковываются.

– d – исключает файл из резервного копирования при использовании утилиты dump.

– e – показывает, что файл использует экстенты в качестве указателей.

– j – для журналируемых файловых систем (ext3, ext4) указывает, что данные файла сначала записываются в журнал, а затем – на диск.

– A – при работе с файлом не обновляется информация о времени последнего доступа.

– D – применяется к каталогам. Изменения в таких каталогах записываются на диск синхронно, минуя кеш.

– S – обеспечивает синхронную запись изменений в файл напрямую на диск, минуя кеш.

Изменить атрибуты файла позволяет команда `chattr`:

```
chattr [модификатор][изменяемые атрибуты] [целевой файл или папка]
```

Таким образом, если необходимо защитить какой-то важный файл от изменений, можно использовать такую команду (рисунок 112):

```
chattr +i example.txt
```

```
rosa-fk7edm user # chattr +i example.txt
rosa-fk7edm user # lsattr example.txt
----i---------e------- example.txt
rosa-fk7edm user # rm -f example.txt
rm: невозможно удалить 'example.txt': Операция не позволена
rosa-fk7edm user # mv example.txt file.txt
mv: невозможно переместить 'example.txt' в 'file.txt': Операция не позволена
rosa-fk7edm user # echo "Hello" > example.txt
bash: example.txt: Операция не позволена
rosa-fk7edm user #
```

Рисунок 112 - Проверка выполнения действий над файлом example.txt

На рисунке 112 видно, что теперь выполнение действий над файлом example.txt невозможно.

Если же нужно вернуть файл в нормальное состояние, необходимо выполнить обратную операцию:

```
chattr -i example.txt
```

Для просмотра более подробной информации о файловых атрибутах, их ограничениях и правилах применения используют команду:

```
man chattr
```

7.1.4.1 SUID (Set User ID)

SUID – это атрибут исполняемого файла, позволяющий запустить его с правами владельца. В ОС РОСА "ХРОМ" приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность, т. к. процесс с правами пользователя не сможет получить доступ на запись к важным системным файлам, например /etc/passwd, который принадлежит суперпользователю root.

Если на исполняемый файл установлен атрибут suid, то при выполнении эта программа автоматически меняет "эффективный userID" на идентификатор того пользователя, который является владельцем этого файла. Таким образом, независимо от того, кто запускает эту программу, она при выполнении имеет права хозяина этого файла.

7.1.4.2 SGID (Set Group ID)

Атрибут SGID аналогичен SUID, но относится к группе. При этом, если для каталога установлен бит SGID, то создаваемые в нем объекты будут получать группу владельца каталога, а не пользователя.

7.1.4.3 UMASK

Любой процесс при создании нового файла определяет, какие права доступа нужно задать для данного файла. Зачастую запрашиваются права "0666" (чтение и запись всеми), что дает больше разрешений, чем необходимо в большинстве случаев. Но при этом каждый раз, когда создается новый файл, Система обращается к параметру, называемому umask. Система использует значение umask, чтобы понизить изначально задаваемые разрешения на более разумные и безопасные. Просмотреть текущие настройки umask можно, набрав в командной строке:

```
user@user:~$ umask
0002
```

В ОС РОСА "ХРОМ" значением по умолчанию для umask является "0022", что позволяет другим пользователям читать новые файлы (если имеется соответствующий доступ), но не изменять их. Чтобы автоматически обеспечивать бóльший уровень защищенности для создаваемых файлов, можно изменить настройки umask:

```
user@user:~$ umask 0077
```

Такое значение umask приведет к тому, что группа и прочие пользователи не будут иметь никаких прав доступа для всех вновь созданных файлов.

В отличие от "обычного" назначения прав доступа к файлу, umask задает, какие права доступа должны быть отключены.

7.1.4.4 Изменение SUID и SGID

Способ установки и удаления битов SUID и SGID:

– чтобы задать бит SUID:

```
chmod u+s <путь к файлу>
```

– чтобы задать бит SGID:

```
chmod g+s <путь к директории>
```

– чтобы удалить бит SUID:

```
chmod u-s <путь к файлу>
```

– чтобы удалить бит SGID:

```
chmod g-s <путь к директории>
```

7.1.5 Расширенные списки контроля доступа (ACL)

ACL (Access Control List, список контроля доступа) – это механизм в ОС, который позволяет более гибко управлять доступом к файлам и папкам, чем это позволяют права доступа по умолчанию. С ACL можно установить права доступа для конкретных пользователей и групп пользователей на конкретные файлы и папки.

ACL обычно состоит из двух типов прав: пользовательские права и групповые права. Пользовательские права позволяют устанавливать права доступа для конкретных пользователей, а групповые права – для групп пользователей.

Преимущество использования ACL заключается в том, что он позволяет установить более точные права доступа, чем это возможно с помощью стандартных прав доступа на основе пользователей и групп. Например, если есть файл, к которому нужен доступ только для нескольких пользователей, можно установить ACL для этого файла, чтобы ограничить доступ только для этих пользователей, не давая права доступа всем остальным пользователям.

Команды для работы с ACL в ОС включают `getfacl` и `setfacl`. Команда `getfacl` используется для просмотра текущих прав доступа ACL на файлы и папки, а команда `setfacl` – для изменения или добавления новых прав доступа ACL на файлы и папки.

Знание ACL может быть полезно для системных администраторов, которые хотят более тонко настраивать доступ к файлам и папкам на своих серверах.

В ОС списки ACL хранятся в расширенных атрибутах (extended attributes, xattr), которые являются дополнительными метаданными, хранящимися в файловой системе. Большинство файловых систем, в т.ч. ext4, BTRFS, XFS, поддерживают хранение ACL в xattr. Однако, не все файловые системы поддерживают ACL и хранение их в xattr. Например, файловая система FAT32 полностью не поддерживает ACL.

Пакет с утилитами для работы с ACL уже предустановлен в ОС. При необходимости его можно установить из репозитория командой:

```
sudo dnf install acl
```

7.1.5.1 Команды getfacl и setfacl

Команда `getfacl` отображает права доступа в ОС в виде ACL:

```
user$ touch файл1
user$ getfacl файл1
# file: файл1
```

```
# owner: kofler
# group: kofler
user::rw-
group::r--
other::r--
user$ ls -l файл1
-rw-r--r-- 1 kofler... файл2
```

С помощью `setfacl` можно определить новые права доступа. Следующие команды предоставляют пользователю gabi и всем членам группы docuteam доступ к файлу с правом читать и изменять этот файл, однако закрывают пользователю kathrin какой-либо доступ к файлу:

```
user$ setfacl -m gabi:rw файл1
user$ setfacl -m g:docuteam:rw файл1
user$ setfacl -m kathrin:- файл1
```

Список прав команды `getfacl` немного длиннее. Теперь команда `ls` показывает на месте прав доступа для членов группы специальную ACL-маску. За буквами, означающими доступ, следует символ "+", указывающий на наличие правил ACL.

```
user$ getfacl файл1
# file: файл1
# owner: kofler
# group: kofler
user::rw-
user:gabi:rw-
user:kathrin:---
group::r--
group:docuteam:rw-
mask::rw-
other::r--
user$ ls -l файл1
-rw-rw-r--+ 1 kofler... файл1
```

Как правило, списки контроля доступа применяются для того, чтобы предоставить определенному пользователю доступ к его файлам, не давая такого доступа всем остальным пользователям. В таком случае потребуется обратиться к администратору, чтобы он создал группу, к которой будет принадлежать сам пользователь и те пользователи, вместе с которыми предполагается обрабатывать файлы. При использовании ACL нужно выполнить команду:

```
setfacl -m пользователь:rw файл
```

7.1.5.2 Маски

Маска ограничивает права, предоставляемые правилами ACL. Если, например, для маски ACL устанавливается значение "r", то ни одно правило не может дать пользователю права изменять или выполнять файлы. Таким образом, ACL-маска имеет приоритет над правилами ACL. Но в любом случае она никак не влияет на обычные права доступа, которые предоставляются пользователю или группе пользователей традиционным способом.

При изменении правила ACL командой `setfacl` маска автоматически рассчитывается заново так, чтобы могли выполняться все прочие правила ACL. Данная маска отображается с помощью команды `getfacl` и также учитывается командой:

```
ls -l
```

Можно явным образом настроить маску с помощью команды:

```
setfacl -m:rwx файл
```

Таким образом, ограничиваются права ACL. Следует учитывать, что маска действует лишь до тех пор, пока не будет определено новое правило ACL. В таком случае маска ACL будет автоматически пересчитана (если только не запрещено с помощью параметра -n).

7.1.5.3 Наследование

При работе с каталогами можно установить второй набор правил для стандартных списков контроля доступа. Стандартные ACL не управляют доступом к каталогу, а служат образцом для новых файлов. Любой файл, создаваемый внутри данного каталога, наследует стандартные списки контроля доступа, указанные для этого каталога. При одновременном применении нескольких списков контроля доступа новый каталог с правильно подобранными стандартными ACL может применяться в работе в качестве отправной точки.

7.1.5.4 Совместимость

Распространение списков контроля доступа сильно осложняется тем, что многие стандартные команды и практически все пользовательские программы просто игнорируют ACL. Если просто скопировать файл с правилами ACL с помощью команды cp, то в скопированном файле правила ACL не сохранятся. То же самое произойдет, если открыть файл в текстовом редакторе OpenOffice или Gimp и сохранить его под другим именем. При использовании команды `cp` будет

полезен параметр -p, но у большинства других команд и программ подобные параметры отсутствуют, либо эти программы вообще не приспособлены к работе с ACL.

Проблемы возникают и при резервном копировании. Команды `tar` и `rsync` элиминируют правила ACL. Файловая система CD и DVD не рассчитана на работу с ACL, так что сохраняемая в них информация теряется.

Есть два выхода: во-первых, можно применять вместо `tar` версию `star`, совместимую с ACL, во-вторых, перед резервным копированием можно создавать отдельный текстовый файл, в который заносятся ACL-правила всех копируемых файлов. После резервного копирования правил ACL восстанавливаются на основании этого файла.

```
getfacl -R --skip-base. > acl-backup (Сохранение ACL-правил)
setfacl --restore=acl-backup (Восстановление ACL-правил)
```

7.1.5.5 Расширенные атрибуты

7.1.5.5.1 Команды setfattr и getfattr

На следующих примерах показано, как сохранять атрибуты с помощью команды `setfattr` и считывать их с использованием `getfattr`:

```
user$ touch ϕайл2
user$ setfattr -n user.language -v ru ϕайл2
user$ setfattr --name=user.charset --value=utf8 ϕайл2
user$ getfattr -d ϕайл2
# file: ϕайл2
user.charset="utf8"
user.language="ru"
```

Команда `getfattr` обычно возвращает только те атрибуты, чье название начинается с user. Если требуется увидеть другие атрибуты, нужно указывать имя атрибута вместе с меткой -n, а образец атрибута с меткой -m.

7.1.5.6 Совместимость с расширенными атрибутами

К сожалению, пока практически не существует программ, которые сохраняли бы расширенные атрибуты при копировании, архивировании и т. д. Даже следующая команда игнорирует атрибуты:

```
cp -p
```

При создании резервных копий лучше всего поступать так же, как и при работе с ACL создавать при копировании текстовый файл со всеми

расширенными атрибутами. На основании этого файла можно потом восстановить расширенные атрибуты:

```
user$ getfattr -R. > ea-backup (Сохранение атрибутов)
user$ setfattr --restore=ea-backup (Восстановление атрибутов)
```

7.1.5.7 Примеры настройки ACL

Типовые примеры настройки списков доступа:

а) Предоставление прав конкретному пользователю:

Чтобы дать пользователю user1 права на чтение и выполнение файла нужно запустить следующую команду:

```
setfacl -m u: user1:rx файл
```

б) Настройка прав для группы editors на запись в каталог:

```
setfacl -m g:editors:rwx каталог
```

в) Удаление записей ACL для файла:

```
setfacl -b файл
```

г) Установка прав по умолчанию для всех новых файлов в каталоге:

```
setfacl -m d:u:john:rwx каталог
```

д) Настройка ACL для существующих файлов. В качестве примера используются следующие исходных данные:

– каталог /media/buh (на поддерживающей ACL файловой системе: ext4, BTRFS, XFS и др.);
– группа пользователей buh (бухгалтерия);
– пользователи ivanova, petrova, sidorova, входящие в группу buh.

Необходимо всех пользователей из группы buh – ivanova, petrova, sidorova – наделись следующими правами:

– читать файлы в /media/buh;
– создавать новые файлы и каталоги;
– изменять существующие файлы.

Для настройки ACL для существующих файлов нужно выполнить следующие действия:

а) Выставить права на каталог /media/buh для группы buh, чтобы ее пользователи могли записывать и читать файлы в этом каталоге:

```
sudo setfacl -R -m g:buh:rwx /media/buh
```

Эта команда добавляет права для группы buh на папку /media/buh и все ее подкаталоги и файлы. Опция -R означает, что изменения должны применяться рекурсивно для всех подкаталогов и файлов. Опция -m означает, что изменяются существующие права. Параметр g:buh:rwx означает, что выставляются права для группы buh на чтение, запись и выполнение.

б) Проверить, что права настроены правильно, можно командой:

```
sudo getfacl /media/buh
```

В выводе на экран должно быть примерно следующее:

```
#file: /media/buh/
# owner: <owner>
# group: buh
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:group:buh:rwx
default:other::r-x
```

В этом выводе для группы buh выставлены права на чтение, запись и выполнение, и группа buh имеет права на чтение и запись файлов в папке /media/buh, все ее подпапки и файлы в них.

в) Автоустановка ACL для создаваемых файлов и папок. Ключ -d в команде `setfacl` позволяет установить ACL по умолчанию на вновь создаваемые файлы и папки в указанной директории.

Для того, чтобы установить права по умолчанию для группы buh на каталог /media/buh, нужно выполнить следующую команду:

```
sudo setfacl -R -d -m g:buh:rwx /media/buh
```

Здесь ключ -d указывает, что устанавливается ACL по умолчанию, а ключ -m указывает, что изменяются существующие права. Аргумент g:buh:rwx задает права для группы buh на чтение, запись и выполнение. Теперь при создании новых файлов и подкаталогов в каталоге /media/buh они будут автоматически получать ACL с правами, выставленными по умолчанию.

Чтобы проверить, что ACL по умолчанию настроены правильно, нужно выполнить команду:

```
sudo getfacl /media/buh
```

В выводе должно быть примерно следующее:

```
# file: /media/buh/
# owner: <owner>
# group: buh
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:group:buh:rwx
default:other::r-x
```

В этом выводе строка default:group:buh:rwx указывает, что группа buh имеет права на чтение, запись и выполнение для всех вновь создаваемых файлов и подкаталогов в каталоге /media/buh.

7.2 Резервное копирование данных

Резервное копирование данных – это важный процесс для обеспечения безопасности и сохранности информации. В Системе для автоматизированного резервного копирования используются утилита rsync и таймеры (файлы с расширением .timer) systemd. Эта комбинация позволяет выполнять регулярные бэкапы файлов и каталогов с минимальной нагрузкой на Систему.

rsync – это мощный инструмент для синхронизации и копирования данных, который поддерживает локальные и удаленные копии, а также предоставляет функции инкрементного копирования. Утилита использует алгоритм, который минимизирует объем передаваемых данных, копируя только измененные файлы или части файлов.

Для организации эффективных процедур резервного копирования и восстановления данных рекомендуется:

– регулярно проверять и обновлять резервные копии;

– хранить резервные копии на разных физических носителях и в удаленных хранилищах;

– периодически тестировать восстановление данных для предотвращения проблем при реальных аварийных ситуациях;

– для хранения архивных копий конфиденциальных данных должны применяться организационные или технические средства предотвращения несанкционированного доступа.

Резервное копирование бывает нескольких видов в зависимости от объема данных, подлежащих сохранению, и особенностей процесса восстановления:

а) Полное резервное копирование (Full Backup). Этот тип копирования включает полное сохранение всех данных, независимо от того, изменялись они с последнего копирования или нет. Преимущества полного резервного копирования заключаются в простоте процесса восстановления, однако его выполнение требует больше времени и места на носителе.

б) Инкрементное резервное копирование (Incremental Backup). Инкрементное копирование сохраняет только те данные, которые изменились после последнего резервного копирования (будь то полное или инкрементное). Это позволяет экономить место и время на создание копий, однако для восстановления всех данных потребуется несколько операций, так как нужно будет применить полную копию и все последующие инкрементные копии.

в) Дифференциальное резервное копирование (Differential Backup). Дифференциальное копирование сохраняет изменения, произошедшие с момента последнего полного резервного копирования. В отличие от инкрементного копирования, дифференциальное требует больше места по мере увеличения объема изменений, но процесс восстановления проще, так как требуется только полная копия и последняя дифференциальная копия.

г) Миррориг (Mirroring). Это процесс создания точной копии данных в реальном времени на другом устройстве или сервере. Обычно используется для высоко доступных систем, где важно иметь постоянно актуальную резервную копию данных.

7.2.1 Планирование резервного копирования и восстановления данных

При создании расписания резервного копирования важно определить, какие данные необходимо копировать, как часто это делать и на каких носителях будут храниться копии. План должен предусматривать возможность восстановления данных за любой период времени, например, для восстановления файла в состоянии, близком к текущему дню.

Для достижения этого могут применяться различные типы резервного копирования. Комбинация инкрементного и дифференциального резервного копирования позволяет достичь баланса между сохранением данных и эффективностью использования ресурсов. Полное копирование фиксирует все файлы системы, а обновляемое копирует только те, которые изменились с момента последнего сохранения.

При использовании многоуровневой схемы резервного копирования, например, полного копирования раз в месяц и инкрементного – раз в день, можно оптимизировать восстановление данных. Такая схема позволяет сократить

время на ежедневное копирование, сохраняя при этом возможность точного восстановления данных.

Важно – Расписание резервного копирования должно быть документировано и доступно пользователям, чтобы обеспечить прозрачность процесса и возможность восстановления.

Рекомендуется использовать схему резервного копирования "день–неделя–месяц" для сохранения трех копий данных (ежедневной, еженедельной и ежемесячной), что позволит избежать перезаписи поврежденных файлов.

План восстановления Системы — неотъемлемая часть плана резервного копирования. В случае аварийной ситуации важно знать, как быстро и полно можно восстановить Систему или отдельные файлы. В рамках плана следует определить местоположение носителей с резервными копиями, их доступность и устойчивость к повреждениям. Также важно предусмотреть, как резервные копии могут быть защищены от сбоев Системы, на которой они хранятся.

Для того чтобы план восстановления оставался надежным, необходимо регулярно проверять исправность носителей с резервными копиями. Это может включать выборочную проверку файлов или полное чтение копий после их создания. Также важно проводить тестирование восстановительных процедур, чтобы удостовериться, что в случае необходимости данные можно будет быстро и корректно восстановить.

Рекомендуется периодически тестировать носители и процедуру восстановления, чтобы убедиться в их работоспособности.

7.2.2 Создание резервных копий

Основные параметры утилиты rsync:

- -a (архивный режим) – сохранение структуры каталогов и прав доступа.
- -v (verbose) – выв подробной информации о процессе копирования.
- --delete – удаление файлов в целевом каталоге, если они были удалены в исходном.
- -z – сжатие файлов при передаче;
- --progress – отображение прогресса копирования.

Пример команды для создания резервной копии каталога /home на внешний диск:

```
rsync -av --delete /home/ /mnt/backup/home/
```

где /mnt/backup/home/ – это целевой каталог, в который будут скопированы данные.

Автоматизация создания резервных копий в ОС осуществляется с использованием systemd. Для автоматического запуска задач резервного

копирования используют планировщик systemd timers, который заменяет классический cron.

Для настройки заданий резервного копирования нужно придерживаться следующей инструкции:

а) Создание unit-файл /etc/systemd/system/backup.service для выполнения задачи резервного копирования:

```
sudo nano /etc/systemd/system/backup.service
```

Содержимое файла должно быть следующим:

```
[Unit]
Description=Резервное копирование каталога /home
Wants=backup.timer

[Service]
ExecStart=/usr/bin/rsync -a --delete /home/ /mnt/backup/home/

[Install]
WantedBy=multi-user.target
```

где в строке Wants= прописывается имя таймер файла, который создается на следующем шаге. А в строке WantedBy= указывается уровень загрузки Системы, при котором будет работать данный .service файл.

б) Создание файла с расширением .timer и названием /etc/systemd/system/backup.timer для запуска сервиса:

```
sudo nano /etc/systemd/system/backup.timer
```

Содержимое файла должно быть следующим (для примера рассмотрим задание, в котором копирование будет запускаться ежедневно):

```
[Unit]
Description=Таймер для запуска резервного копирования
Requires=backup.service

[Timer]
OnCalendar=daily
Persistent=true
Unit=backup.service

[Install]
WantedBy=timers.target
```

где в строке Requires= указывается имя созданного выше .service файла.и регламентируется то, что таймер запускается только при его наличии. И в строке Unit= также указывается имя созданного файла .service который будет запускаться при срабатывании таймера.

Параметр Persistent=true должен быть включен в содержание файла, т.к. он позволяет выполнять задачу после пропуска, если Система была выключена.

в) Запуск и активация таймера:

```
sudo systemctl start backup.timer
sudo systemctl enable backup.timer
```

Теперь резервное копирование будет выполняться автоматически согласно заданному расписанию.

Подробно о создании и настройке расписания для файла .timer systemd см. раздел 3.4.2.1.

Дополнительные параметры для более гибкой настройки резервного копирования:

а) Инкрементное резервное копирование, которое позволяет копировать только изменённые файлы. Для этого используется флаг --link-dest. Для проведения инкрементного копирования нужно настроить systemd сервис и таймер-файлы следующим образом:

```
[Service]
ExecStart=/usr/bin/rsync -av --link-dest=/mnt/backup/home/previous /home/ /mnt/backup/home/current/
```

Файл .timer, например, можно настроить для ежедневного инкрементного резервного копирования в 1:00:

```
[Timer]
OnCalendar=*-*-* 01:00:00
Persistent=true
```

б) Ограничение пропускной способности при резервном копировании. Чтобы предотвратить перегрузку сети, можно ограничить скорость передачи данных при проведении резервного копирования. Для этого в файле .service нужно указать следующие параметры:

```
[Service]
ExecStart=/usr/bin/rsync -av --bwlimit=5000 /home/ /mnt/backup/home/
```

где 5000 означает ограничение в 5000 КБ/с (5 МБ/с). Это полезно для резервного копирования в рабочее время, когда важно сохранять пропускную способность сети.

в) Задание для копирования сжатых файлов с использованием опции -z:

```
[Service]
ExecStart=/usr/bin/rsync -avz /home/ /mnt/backup/home/
```

Дополнительно должен быть также задан .timer файл для данного задания.

г) Настройка уведомления по электронной почте о завершении задания через systemd service:

```
[Service]
ExecStart=/usr/bin/rsync -av --delete /home/ /mnt/backup/home/
ExecStartPost=/usr/sbin/sendmail admin@example.com < /var/log/backup.log
```

Дополнительно должен быть также задан .timer файл для данного задания.

д) Настройка резервного копирования с условием запуска задания только при подключении внешнего диска:

```
[Service]
ExecStart=/bin/bash -c 'if mount | grep /mnt/backup; then rsync -av --delete /home/ /mnt/backup/home/; fi'
```

Дополнительно должен быть также задан .timer файл для данного задания.

е) Комплексное задание с ограничением скорости, сжатием и уведомлением по электронной почте:

```
[Service]
ExecStart=/usr/bin/rsync -avz --bwlimit=5000 --delete /home/ /mnt/backup/home/
ExecStartPost=/usr/sbin/sendmail admin@example.com < /var/log/backup.log
```

Дополнительно должен быть также задан .timer файл для данного задания.

ж) Отчёт по результатам выполнения задания. Чтобы сохранить лог выполнения задания в отдельный файл, можно добавить следующую команду:

```
[Service]
```

```
    ExecStart=/usr/bin/rsync -av --delete /home/ /mnt/backup/home/
    StandardOutput=append:/var/log/backup.log
```

Логи будут записываться в файл /var/log/backup.log после каждого выполнения задания.

з) Для контроля процесса резервного копирования и проверки резервных копий можно проверять логи systemd:

```
    journalctl -u backup.service
```

Для проверки целостности данных можно использовать флаг --dry-run:

```
    rsync -av --dry-run /home/ /mnt/backup/home/
```

Флаг --dry-run позволяет симулировать процесс копирования без фактического изменения данных, чтобы увидеть, какие файлы будут обновлены.

7.2.3 Восстановление данных из резервной копии

Восстановление данных происходит обратным копированием из резервного хранилища. Для примера рассмотрим копирование данных каталога /home:

```
    rsync -av /mnt/backup/home/ /home/
```

Этот процесс гарантирует восстановление всей информации в ее актуальном виде.

7.2.4 Архивирование

Архивирование в ОС – это то же самое, что и сжатие файлов.

Архивирование – это объединение нескольких небольших файлов в один с целью более удобной последующей передачи, хранения, шифрования или сжатия. Архивация выполняется специальными утилитами.

Самой популярной для ОС утилитой для архивации является `tar`. Она используется почти всегда для архивации исходников, упаковки пакетов. Для сжатия используются другие утилиты, в зависимости от алгоритма сжатия, например: zip, bz, xz, Lzma и т д. Сначала выполняется архивация, затем сжатие отдельными программами. Автоматический запуск некоторых утилит сжатия для только что созданного архива поддерживается в `tar` и других подобных программах с помощью специальных опций.

Также полезной возможностью архивации является шифрование.

7.2.4.1 Утилита Tar

`Tar` – это стандартная утилита, с помощью которой выполняется архивирование файлов ОС. Постепенно из небольшой программы архивации она превратилась в мощный инструмент, поддерживающий работу со многими вариантами архивов и алгоритмами сжатия. Программа поддерживает большое количество параметров.

Общий вид команды:

```
$ tar опции файл_для_записи /папка_файлами_для_архива
```

Основные опции:

– A – добавить файл к архиву;
– c – создать архив;
– d – сравнить файлы архива и распакованные файлы в ФС;
– j – сжать архив с помощью Bzip;
– z – сжать архив с помощью Gzip;
– r – добавить файлы в конец архива;
– t – показать содержимое архива;
– u – обновить архив относительно ФС;
– x – извлечь файлы из архива;
– v – показать подробную информацию о процессе работы;
– f – файл для записи архива;
– -C – распаковать в указанную папку;
– --strip-components – отбросить n вложенных папок.

Чтобы создать архив, используют такую команду:

```
tar -cvf archive.tar.gz /PATH/to/files
```

Чтобы распаковать архив `tar`:

```
tar -xvf archive.tar.gz
```

Сжатый архив создается точно так же, только с опцией -z, с шифрованием gizp; если нужно bzip, то применяется опция -j:

```
tar -zcvf archive.tar.gz /PATH/to/files
$ tar -zxvf archive.tar.gz
```

Например, чтобы заархивировать папку в ОС:

```
tar -zcvf home.tar.gz ~/
```

По-другому тот же архив можно получить, если сначала создать обычный архив с помощью `tar`, а потом сжать его утилитой для сжатия, при этом остается больше контроля над процессом сжатия:

```
Gzip archive.tar
```

Также можно убрать сжатие:

```
gunzip archive.tar.gz
```

Чтобы добавить файл в архив, используют:

```
tar -rvf archive.tar file.txt
```

Для извлечения одного файла синтаксис тот же:

```
tar -xvf archive.tar file.txt
```

Можно извлечь несколько файлов по шаблону соответствия с помощью параметра wildcard, например, извлечь все php-файлы:

```
tar -xvf archive.tar --wildcards '*.php'
```

По умолчанию распаковать архив tar можно в текущую папку с именем архива, а чтобы распаковать в нужную папку используют ключ -C:

```
tar -xvf archive.tar -C /PATH/to/dir
```

Далее рассматриваются альтернативы утилиты `tar`.

7.2.4.2 Утилита Shar

Утилита Shar позволяет создавать самораспаковывающиеся архивы. По сути, это скрипт оболочки, и для распаковки ему необходима оболочка Bash или другая совместимая с Bourne Shell. У Shar есть несколько преимуществ, но также она является потенциально небезопасной, так как архив представляет собой исполняемый файл.

Опции Shar:

– -o – сохранять архив в файл вместо стандартного вывода;
– -l – ограничить размер выходного файла;
– -L – ограничить размер выходного файла и разбить его на части;
– -n – имя архива не будет включено в заголовок;
– -a – разрешить автоматическую генерацию заголовков.

Примеры использования Shar для того, чтобы заархивировать папку:

Создать Shar-архив:

```
Shar file_name.extension > filename.Shar
```

Распаковывать Shar-архив:

```
./filename.Shar
```

7.2.4.3 Утилита Ar

`Ar` – утилита для создания и управления архивами. В основном используется для архивации статических библиотек, но может быть использована для создания любых архивов. Раньше использовалась довольно часто, но была вытеснена утилитой `tar`. Сейчас используется только для создания и обновления файлов статических библиотек.

Опции `Ar`:

– -d – удалить модули из архива;
– -m – перемещение членов в архиве;
– -p – напечатать специфические члены архива;
– -q – быстрое добавление;
– -r – добавить члена к архиву;
– -s – создать индекс архива;
– -a – добавить новый файл к существующему архиву.

Создать статическую библиотеку libmath.a из объектных файлов substraction.o и division.o:

```
ar cr libmath.a substraction.o division.o
```

Далее необходимо извлечь файлы из архива:

```
ar x libmath.a
```

Таким образом можно распаковать любую статическую библиотеку.

7.2.4.4 Утилита Cpio

Улитита `cpio` (copy in and out – скопировать ввод и вывод) – это еще один стандартный архиватор для ОС. Активно используется для создания initramfs. Архивация в ОС для обычных файлов с помощью этой программы не применяется.

Опции утилиты:

– -a – сбросить время обращения к файлам после их копирования;
– -A – добавить файл;
– -d – создать каталоги при необходимости.

Создать cpio-архив:

```
ls
file1.o file2.o file3.o
ls | cpio -ov > /PATH/to/output_folder/obj.cpio
```

Чтобы распаковать архив:

```
cpio -idv < /PATH/to folder/obj.cpio
```

7.2.5 Сжатие

7.2.5.1 Утилита Gzip

В качестве стандартной утилиты сжатия в ОС чаще всего применяется утилита `Gzip`. Для декомпрессии используются команды `gunzip` или `Gzip -d`.

Общий вид утилиты:

```
$ Gzip опции файл
$ gunzip опции файл
```

Опции:

– -c – выводить архив в стандартный вывод;
– -в – распаковать;
– -f – принудительно распаковывать или сжимать;
– -l – показать информацию об архиве;
– -r – рекурсивно перебирать каталоги;
– -0 – минимальный уровень сжатия;
– -9 – максимальный уровень сжатия.

Например, выполнить сжатие файла:

```
Gzip -c файл > архив.gz
```

А теперь распаковать:

```
gunzip -c архив.gz
```

Но чтобы сжать папку в ОС, придется сначала заархивировать ее с помощью tar, а уже потом сжать файл архива с помощью Gzip.

7.2.5.2 Утилита Bzip

bzip2 – еще одна альтернативная утилита сжатия для ОС. Она более эффективная чем Gzip, но работает медленнее. Для распаковки используют утилиту bunzip2.

Опции утилиты bzip2 аналогичны опциям Gzip.

Чтобы создать архив, используют:

```
bzip2 file
```

В текущем каталоге будет создан файл file.bz2.

7.2.5.3 Утилита Lzma

Lzma – новый и высокоэффективный алгоритм сжатия. Синтаксис и опции тоже похожи на Gzip.

Для распаковки используют команду:

```
unLzma
```

Наиболее часто используемые параметры утилиты `lzma`:

– -z – выполнить сжатие указанного файла (по умолчанию);
– -d – распаковать ранее сжатый файл;
– -k – сохранить исходный файл после сжатия или распаковки;
– -c – вывести результат сжатия или распаковки в стандартный поток вывода (stdout), не создавая новых файлов;
– -f – принудительно перезаписать существующий файл без подтверждения;
– -v – отобразить подробную информацию о процессе сжатия или распаковки;
– -t – проверить целостность сжатого файла без его распаковки;
– -0 ... -9 – установить уровень сжатия: от минимального (-0, быстрое сжатие) до максимального (-9, более эффективное, но медленное);
– --format=xz – использовать формат XZ вместо стандартного LZMA;
– --keep – аналог опции -k, сохраняет исходный файл после обработки;
– --stdout – аналог опции -c, направляет результат в стандартный вывод;
– --threads=N – задать количество потоков при многопоточном сжатии (актуально для современных версий).

7.2.5.4 Утилита Xz

Утилита `xz` используется для сжатия и распаковки файлов с использованием алгоритма LZMA2. Формат сжатия .xz отличается высокой степенью сжатия и эффективным использованием ресурсов, при этом полностью совместим с форматом .lzma.

Наиболее часто используемые параметры:

– -d – распаковывает указанный файл.
– -k – сохраняет исходный файл после сжатия или распаковки.
– -t – выполняет проверку целостности сжатого архива без его распаковки.
– -v – включает подробный вывод информации о ходе выполнения операции.

– -0 ... -9 – задаёт уровень сжатия: от минимального (-0, быстрее, но менее эффективно) до максимального (-9, медленнее, но с наибольшим коэффициентом сжатия).

– --format=lzma – указывает использовать формат LZMA вместо XZ.

– --threads=N – задаёт количество потоков для многопоточного сжатия (рекомендуется использовать значение "--threads=auto" для автоматического определения числа доступных ядер процессора).

Для сжатия файла применяют команду вида:

```
xz файл
```

После выполнения команда создаёт сжатую версию файла с расширением .xz и удаляет оригинал (если не указано иное).

Для сжатия без удаления исходного файла:

```
xz -k файл
```

Опция -k (keep) сохраняет исходный файл после сжатия.

Сжатие с заданием уровня:

```
xz -9 файл
```

Параметр -9 устанавливает максимальный уровень сжатия (по умолчанию используется -6).

Для сжатие нескольких файлов в архив

```
tar -cJf архив.tar.xz каталог/
```

или распаковка:

```
tar -xJf архив.tar.xz
```

Здесь параметр -J указывает на использование xz в составе `tar`.

Практические рекомендации по использованию xz:

– для больших файлов рекомендуется использовать параметр --threads=auto для многопоточного сжатия;

– при архивации целых каталогов удобнее использовать комбинацию `tar -cJf`, так как xz работает только с отдельными файлами;

– для резервного копирования системных файлов оптимально использовать средний уровень сжатия (-6) для баланса между скоростью и эффективностью.

7.2.5.5 Утилита Zip

`Zip` – кроссплатформенная утилита для создания сжатых архивов формата zip, совместимая с Windows-реализациями этого алгоритма. Zip-архивы очень часто используются для обмена файлами в сети Интернет. С помощью этой утилиты можно сжимать как файлы, так и папки.

Синтаксис утилиты:

```
$ zip опции файлы
$ unzip опции архив
```

Опции утилиты:

– -d – удалить файл из архива;
– -r – рекурсивно обходить каталоги;
– -0 – только архивировать, без сжатия;
– -9 – наилучшая степень сжатия;
– -F – исправить zip-файл;
– -e – шифровать файлы.

Чтобы создать Zip-архив, используют:

```
zip -r /PATH/to/files/*
```

Для распаковки:

```
unzip archive.zip
```

7.2.5.6 Утилита RAR

Для работы с RAR-архивами используют утилиты `rar` для создания архивов и `unrar` для распаковки. Установить архиватор можно средствами стандартного пакетного менеджера ОС, но для установки последней актуальной версии используют другой способ:

а) скачать с официального сайта проекта www.rarlab.com архив с бинарными файлами и извлечь из него файлы rar и unrar; скопировать их, например, в /usr/bin/ либо произвести установку:

```
wget
tar xzf rar linux-x64-612.tar.gz
cd rar
make install
```

б) команда `make install` выполнит следующие действия согласно содержимому файла makefile:

```
mkdir -p /usr/local/bin
mkdir -p /usr/local/lib
cp rar unrar /usr/local/bin
cp rar files.lst /etc
cp default.sfx /usr/local/lib
```

Далее приводятся основные команды `rar`.

Создание архива файлов file01.sql, file02, file03:

```
rar a archive.rar file01.sql file02 file03
```

Создать архив директории dir01 можно рекурсивно:

```
rar -r a archive.rar dir01
```

Добавить файлы в архив можно с помощью ключа u, а удалить из архива ключом d:

```
rar u archive.rar file02
rar d archive.rar file02
```

Также можно указать степень сжатия архива. Параметр сжатия принимает значения от 0 до 5, при этом 0 – без использования сжатия, по умолчанию используется степень 3:

```
rar a -m5 archive.rar file02
```

Для распаковки архива также можно использовать команду `rar` с опцией x:

```
rar x archive.rar
```

Для распаковки RAR-архивов используется утилита `unrar`. Распаковать архив:

```
unrar x archive.rar
```

Распаковать архив в определенную директорию:

```
unrar x archive.rar ./dir
```

Проверить, что архив целый:

```
unrar t archive.rar
```

Просмотреть список файлов в архиве:

```
unrar l archive.rar
```

7.3 Контроль целостности Системы

7.3.1 Контроль целостности с помощью AIDE

7.3.1.1 Общие принципы работы AIDE

AIDE – программа контроля целостности Системы – используется для защиты от вредоносных программ, вирусов и обнаружения несанкционированных действий. AIDE создает базу данных с информацией о файлах и каталогах (контрольная сумма, время изменения) и периодически сравнивает их текущее состояние с этой базой.

Это позволяет выявить нарушения целостности Системы и принять меры для восстановления измененных файлов из резервной копии или с образа восстановления Системы. AIDE помогает сократить время расследования инцидентов, сосредоточившись на файлах, которые были изменены.

Дополнительные возможности AIDE:

– поддержка различных атрибутов файлов, в том числе: тип файла, inode, uid, gid, разрешения, количество ссылок, mtime, ctime и atime;

– поддержка сжатия Gzip, SELinux, XAttrs, Posix ACL и атрибутов файловой системы;

– поддержка различных алгоритмов, включая md5, sha1, sha256, sha512, rmd160, crc32 и т. д.;

– отправка уведомлений по электронной почте.

7.3.1.2 Установка и создание базы данных

Пользователю необходимо установить пакеты AIDE из репозиториев в Систему с помощью команды:

```
sudo dnf install aide
```

Далее необходимо создать базу данных AIDE, которая будет хранить контрольные суммы, права доступа и другие атрибуты файлов и директорий. Сравнивая текущие данные с ранее созданной базой, администратор может легко обнаружить любые отклонения, что позволяет оперативно реагировать на подозрительные изменения.

База данных в AIDE создается с помощью команды:

```
sudo aideinit
```

В ходе создания БД будет запрошен пароль от GPG-ключа, которым будет подписываться база AIDE для контроля ее целостности. Затем будет выполнено первичное заполнение базы /var/lib/aide/aide.db, т.е. зафиксировано заведомо

корректное состояние Системы. Проверка целостности ОС и обновление базы данных.

Также необходимо своевременно проводить обновление базы данных AIDE после внесения легитимных изменений в Систему (например, при установке обновлений), чтобы актуализировать её состояние. Для этого выполните команду:

```
sudo aide --update
```

Затем замените старую базу на новую:

```
mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

7.3.1.3 Настройка конфигурационного файла

Для проведения более корректных проверок AIDE необходимо правильно настроить его конфигурационный файл, который находится по пути /etc/aide.conf. Этот файл определяет, какие файлы и каталоги будут проверяться, какие атрибуты файлов отслеживаются, а также исключения из проверки. Вот основные шаги по настройке конфигурации AIDE:

Чтобы изменить настройки AIDE, нужно открыть конфигурационный файл в текстовом редакторе:

```
sudo nano /etc/aide.conf
```

Далее задать необходимые атрибуты для проверки в тексте файла.

Для проверки всех атрибутов файла можно использовать следующую запись:

```
/etc p+i+n+u+g+s+m+c+md5
```

Для мониторинга только изменения прав доступа и содержания файлов:

```
/var/log p+c+md5
```

Проверка только атрибутов и прав доступа:

```
NORMAL = p+i+n+u+g+s+m+S+sha256
```

Проверка всех атрибутов, включая контрольные суммы:

```
EXTENDED = p+i+n+u+g+s+m+S+sha256+md5+rmd160
```

где параметры:

– p – права доступа;
– i – inode номер;
– n – количество ссылок;
– u – идентификатор пользователя (UID);
– g – идентификатор группы (GID);

– s – размер файла;
– m – время последней модификации;
– S – время последнего изменения статуса файла;
– sha256, md5 – контрольные суммы файлов.

В отдельном разделе конфигурации также указываются файлы и каталоги, которые будут отслеживаться на предмет изменений. Пример конфигурации:

Проверка корневой файловой системы с полными атрибутами, определёнными в строчке NORMAL:

```
/    NORMAL
```

Проверка каталога /etc с расширенными атрибутами, определенными в строчке EXTENDED:

```
/etc EXTENDED
```

Чтобы избежать проверки определённых системных файлов или временных данных, вы можете добавлять исключения с помощью символа "!" перед каталогом или файлом.

```
!/tmp
!/var/tmp
```

После внесения всех необходимых изменений, нужно сохранить файл и закрыть редактор.

После настройки конфигурации можно протестировать её работу, запустив создание новой базы данных AIDE:

```
sudo aideinit
```

После этого AIDE создаст базу данных, которую можно будет использовать для последующих проверок.

Пример содержимого конфигурационного файла AIDE:

```
database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new

NORMAL = p+i+n+u+g+s+m+S+sha256

/etc    EXTENDED
/bin    NORMAL
/sbin   NORMAL

!/proc
!/sys
```

```
!/var/tmp
!/var/cache
```

Этот пример включает основные правила для мониторинга критически важных директорий Системы и исключает временные и кешированные файлы.

7.3.1.4 Проверка целостности

После создания базы данных и настройки конфигурации можно запустить проверку целостности Системы. Это делается командой:

```
aide --check
```

Команда сравнит текущее состояние файлов с данными, сохраненными в базе. Результаты проверки будут выведены в терминал, где можно увидеть, какие файлы были изменены, добавлены или удалены.

Для запуска проверки немедленно можно воспользоваться командой:

```
sudo systemctl restart aidecheck.service
```

Результаты проверок также записываются в системный журнал, который можно открыть и просмотреть по команде:

```
sudo journalctl -u aidecheck.service
```

При необходимости фиксации корректных изменений конфигурации ОС нужно использовать следующую команду, при этом будет запрошен пароль доступа к ключу GPG:

```
sudo aideupdate
```

Чтобы посмотреть статус последней проверки, нужно выполнить команду:

```
systemctl status aidecheck.service
```

7.3.1.5 Автоматизация проверки

Для удобства администрирования Системы проверку целостности ОС можно автоматизировать с помощью использования таймера systemd. Подробная настройка файлов с расширением .timer рассмотрена в разделе 3.4.2.1.

Для настройки таймера через systemd выполняются следующие шаги для настройки задания:

а) Создание Unit-файла для автоматизации проверки AIDE, где будет указано, как именно systemd должен запускать проверку. Необходимо создать файл /etc/systemd/system/aide-check.service со следующим содержанием:

```
[Unit]
Description=AIDE integrity check
Documentation=man:aide(1)

[Service]
Type=oneshot
ExecStart=/usr/sbin/aide --check
```

б) Далее создать таймер-файл, который будет запускать проверку по расписанию. Например, проверка может выполняться ежедневно. Создайте файл /etc/systemd/system/aide-check.timer со следующим содержанием:

```
[Unit]
Description=Daily AIDE integrity check

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

Таймер можно настроить на любое другое расписание, например, использовать параметр OnCalendar=weekly для еженедельной проверки.

в) Далее необходимо активировать таймер-файл командой:

```
sudo systemctl enable aide-check.timer
sudo systemctl start aide-check.timer
```

Команда `enable` добавляет таймер в автозапуск, а команда `start` запускает таймер сразу после выполнения команды.

г) Для того, чтобы провести проверку статуса таймера и убедиться, что он работает корректно, используется команда:

```
sudo systemctl status aide-check.timer
```

7.3.2 Контроль целостности на основе Afick

7.3.2.1 Назначение и общие сведения

Afick (Another File Integrity CheckKer) является инструментом проверки целостности, который создаёт базу эталонных значений и сравнивает её с текущим состоянием файловой системы.

В отличие от AIDE, Afick работает быстрее на больших файловых деревьях, использует внутреннюю оптимизацию поиска изменений и обеспечивает более гибкое управление конфигурацией. Принцип работы основан на трёх этапах:

а) создание базы контрольных значений;

б) сравнение текущего состояния файлов с эталонной базой;

в) формирование отчёта с указанием новых, изменённых и удалённых объектов.

Afick применяется для обнаружения подозрительных модификаций, возникающих в результате ошибок конфигурации, действий пользователей или возможной компрометации Системы. Инструмент рекомендуется использовать как дополнение к другим механизмам безопасности ОС.

7.3.2.2 Начало работы

Установка выполняется администратором из репозитория ОС:

```
dnf install afick
```

После установки в систему добавляются служебные файлы, утилита проверки и конфигурационный файл /etc/afick.conf.

Файлы Afick размещаются в следующих каталогах:

– основной конфигурационный файл:

```
/etc/afick.conf
```

– каталоги служебных данных:

```
/var/lib/afick/
```

– файлы журналов Afick (если включено логирование):

```
/var/log/afick/
```

Для выполнения операций Afick требуются административные права. Все операции создания или обновления базы выполняются только от имени суперпользователя, так как инструмент должен иметь доступ ко всем контролируемым файлам.

Конфигурационный файл /etc/afick.conf определяет, какие каталоги и файлы подлежат проверке, какие параметры целостности анализируются, а также где хранится база контрольных значений. Файл содержит следующие основные типы директив:

– Общие параметры, которые задают расположение рабочих файлов Afick:

```
database=/var/lib/afick/afick.db
```

```
logfile=/var/log/afick/afick.log
```

– Списки контролируемых файлов и каталогов:

```
include=/etc
include=/usr/bin
include=/usr/sbin
```

По умолчанию включаются основные системные каталоги.

– Исключения из проверок:

```
exclude=/var/log
exclude=/run
exclude=/tmp
exclude=/proc
```

Рекомендуется исключать каталоги, которые меняются регулярно и не имеют отношения к контролю целостности.

– Определение конкретных атрибутов для проверки:

– размер файла (size);

– контрольные суммы (sha1, md5 — в зависимости от выбранного алгоритма);

– права доступа (mode);

– владелец и группа (uid, gid);

– временные метки изменения файла (mtime);

Пример использования всех параметров вместе:

```
check=sha1,size,mtime,uid,gid,mode
```

– Настройка уровней отчётности. Директива имеет вид:

```
report=<уровень>
```

Поддерживаются следующие уровни отчётности:

– short — краткий отчёт; выводится только сводная информация без перечисления конкретных файлов. Используется для минимального контроля в автоматизированных сценариях;

– changes — отчёт только о фактически изменённых объектах; отображаются новые, изменённые и удалённые файлы. Рекомендуемый режим для регулярных проверок;

– all — полный отчёт по всем проверяемым объектам; выводится состояние каждого файла независимо от того, произошло изменение или нет. Применяется для первичного аудита либо детального анализа;

– debug — расширенный отчёт в диагностическом режиме; включает служебную информацию, подробные сообщения о ходе проверки и обработки файлов. Используется для отладки конфигурации.

7.3.2.3 Создание базы целостности и выполнение проверки

После настройки файла /etc/afick.conf требуется создать эталонную базу контрольных значений. Инициализация базы выполняется командой:

```
afick --init
```

В ходе операции Afick анализирует указанные каталоги, вычисляет контрольные суммы, фиксирует атрибуты файлов и сохраняет эталонное состояние в базе данных, расположенной по пути, определённому в конфигурации (по умолчанию /var/lib/afick/afick.db).

Инициализация выполняется один раз — перед началом эксплуатации Системы или после масштабных обновлений.

База Afick содержит следующие сведения о каждом проверяемом файле:

– контрольные суммы;
– права доступа;
– владельца и группу;
– размеры;
– временные метки.

При выполнении последующих проверок Afick сравнивает фактическое состояние системы с данными из базы и фиксирует:

– новые файлы;
– удалённые файлы;
– изменённые файлы;
– изменения атрибутов.

После законного изменения конфигурации (обновление пакетов, изменение системных файлов) база должна быть обновлена с помощью команды:

```
afick --update
```

В противном случае Afick будет продолжать считать данные изменения нарушением целостности.

Рекомендуется выполнять обновление базы только после проверки, что изменения являются корректными и одобрены администратором.

Полная проверка целостности выполняется командой:

```
afick --check
```

По результатам формируется отчёт, в котором изменения классифицируются как:

– added — файл добавлен;
– removed — файл удалён;
– changed — изменено содержимое;
– attrs — изменены атрибуты (права, владелец, размер);
– mtime — изменена временная метка.

Файл отчёта создаётся в соответствии с настройками конфигурации (обычно /var/log/afick/afick.log).

Для обеспечения регулярного контроля целостности рекомендуется автоматизировать выполнение команды проверки с помощью планировщика systemd timers, который заменяет классический cron.

Эталонная база (afick.db) и файл конфигурации должны храниться на носителе с ограниченным доступом. После завершения настройки их можно защитить от модификации, установив неизменяемый атрибут:

```
chattr +i /etc/afick.conf
chattr +i /var/lib/afick/afick.db
```

При использовании Afick на серверах с регулярными изменениями следует корректно поддерживать список исключений, чтобы исключить регулярные изменения служебных каталогов из проверки.

7.4 Формирование замкнутой программной среды

7.4.1 Общие принципы

Замкнутая программная среда (далее – ЗПС) – такая организация ОС, при которой возможен запуск только доверенных приложений, а сторонние скрипты или приложения выполняться не будут. В ОС РОСА "ХРОМ" ЗПС реализуется посредством технологии IMA – путем подписывания всех исполняемых файлов Системы с помощью механизма электронной подписи.

Электронная подпись основана на двух ключах – закрытом, с помощью которого возможно подписать файлы, и открытом, необходимом для проверки подписанного. Специальным образом сконфигурированная Система при попытке запуска любого исполняемого файла или при загрузке библиотеки будет проверять с помощью открытого ключа их подлинность и в случае успеха разрешать запуск.

Следует обратить внимание, что доступ к закрытому ключу должен быть только у уполномоченных лиц, отвечающих за безопасность Системы. Закрытый

ключ не должен храниться на компьютере, где осуществлялось подписывание файлов с его помощью.

Для работы с ЗПС в ОС должны быть установлены пакеты ima-evm-utils, libressl, ima-inspect.

7.4.2 Создание ключей

Для создания открытых и закрытых ключей выполняют следующие действия:

а) Открыть консоль администратора командой (с вводом пароля):

```
sudo -i
```

б) Создать временную папку для ключей и перейти в неё:

```
mkdir ~/ima
cd ~/ima
```

в) Создать файл конфигурации x509.conf, например, командой:

```
nano x509.conf
```

г) Задать содержимое файла следующими настройками:

```
[ req ]
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = IMA
CN = Executable Signing Key
emailAddress = admin@example.tld

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
```

Значения полей "O", "CN", "emailAddress" являются пользовательскими, нужно заменить их на необходимые;

д) Сохранить файл (если используется редактор nano, то последовательно нажать клавиши `Ctrl+O`, `Enter` и `Ctrl+X`);

е) Создать пару ключей командой:

```
libressl req -new -nodes -utf8 -batch -newkey gost2001 -pkeyopt dgst:streebog512 -pkeyopt paramset:A -streebog512 -days 109500 -x509 -config x509.conf -outform DER -out ima_cert.der -keyout ima_priv.pem
```

где:

–ima_cert.der – открытый ключ;

–ima_priv.pem – закрытый ключ.

ж) Создать папку для открытых ключей ядра, скопировать туда открытый ключ:

```
mkdir -p /etc/keys/ima
cp -v ima_cert.der /etc/keys/ima/ima_cert.der
```

Дополнительно нужно скопировать открытый ключ для работы механизма проверки:

```
cp -v ima_cert.der /etc/keys/x509_evm.der
```

7.4.3 Создание политики IMA

Для создания политики IMA, продолжая работать в консоли от имени root, необходимо редактировать файл политики IMA:

```
sudo nano /etc/sysconfig/ima-policy
```

Далее нужно скопировать в него следующее содержимое:

```
# PROC_SUPER_MAGIC
dont_measure fsmagic=0x9fa0
dont_appraise fsmagic=0x9fa0
# SYSFS_MAGIC
dont_measure fsmagic=0x62656572
dont_appraise fsmagic=0x62656572
# DEBUGFS_MAGIC
dont_measure fsmagic=0x64626720
dont_appraise fsmagic=0x64626720
# TMPFS_MAGIC
dont_measure fsmagic=0x01021994
dont_appraise fsmagic=0x01021994
```

```
# RAMFS_MAGIC
dont_appraise fsmagic=0x858458f6
# DEVPTS_SUPER_MAGIC
dont_measure fsmagic=0x1cd1
dont_appraise fsmagic=0x1cd1
# BINFMTFS_MAGIC
dont_measure fsmagic=0x42494e4d
dont_appraise fsmagic=0x42494e4d
# SECURITYFS_MAGIC
dont_measure fsmagic=0x73636673
dont_appraise fsmagic=0x73636673
# SELINUX_MAGIC
dont_measure fsmagic=0xf97cff8c
dont_appraise fsmagic=0xf97cff8c
# CGROUP_SUPER_MAGIC
dont_measure fsmagic=0x27e0eb
dont_appraise fsmagic=0x27e0eb
# CGROUP2_SUPER_MAGIC
dont_measure fsmagic=0x63677270
dont_appraise fsmagic=0x63677270
# NSFS_MAGIC
dont_measure fsmagic=0x6e736673
dont_appraise fsmagic=0x6e736673
measure func=MMAP_CHECK mask=MAY_EXEC
measure func=BPRM_CHECK mask=MAY_EXEC
appraise func=BPRM_CHECK mask=MAY_EXEC appraise_type=imasig
appraise func=MMAP_CHECK mask=MAY_EXEC appraise_type=imasig
```

Затем необходимо сохранить файл и выставить на него права командой:

```
chmod 0400 /etc/sysconfig/ima-policy
```

7.4.4 Подписывание файлов

Для подписания всех исполняемых файлов в Системе, в том числе скриптов и разделяемых библиотек *.so*, необходимо:

а) Создать файл скрипта для их подписи:

```
nano sign.sh
```

б) Скопировать нижеследующий текст в этот файл:

```
    #!/bin/bash
set -xe

    FS="$(findmnt --output FSTYPE / | tail -n1)"
    echo "$FS"
    echo > failed.log

    find / -fstype "$FS" -type f -executable | sort -u | while
read -r line
    do
     if   !   evmctl   ima_sign   --hashalgo   streebog512   --key
ima_priv.pem "$line" ; then
     echo "$line" >> failed.log
     fi
    done

    for D in /lib /lib64 /usr/lib /usr/lib64
    do
     find "$D" -fstype $FS -\! -executable -type f -name "*.so*"
| sort -u | while read -r line
    do
     if !   evmctl   ima_sign   --hashalgo   streebog512   --key
ima_priv.pem "$line"; then
     echo "$line" >> failed.log
     fi
     done
    done
```

в) Сохранить файл скрипта и сделать его исполняемым:

```
    chmod 755 sign.sh
```

г) Запустить скрипт и подписать файлы:

```
    ./sign.sh
```

После запуска (скрипт будет работать некоторое время) можно посмотреть журнал ошибок командой (он должен быть пуст):

```
    cat failed.log
```

7.4.5 Подготовка файловой системы

Если используется ФС ext4, то необходимо добавить опцию монтирования. Для этого нужно открыть файл настроек монтирования командой:

```
sudo nano /etc/fstab
```

Далее нужно добавить в строках с ext4 опцию iversion через запятую после defaults, как показано ниже (рисунок 113), после этого сохранить файл.

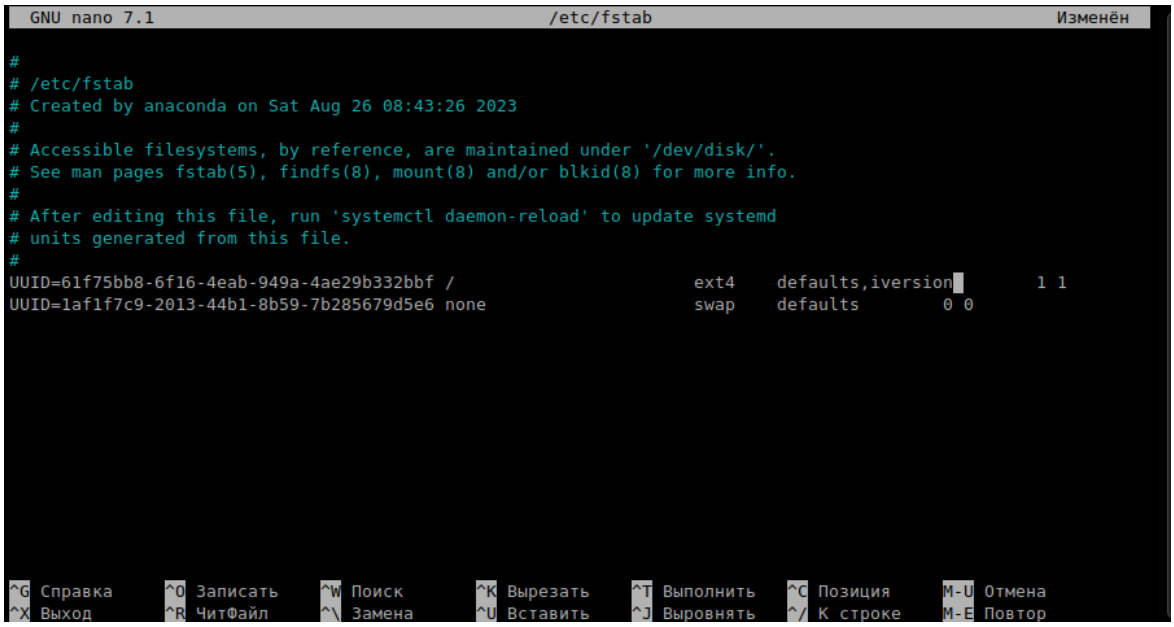


Рисунок 113 – Добавление опции монтирования для работы IMA

7.4.6 Добавление в образ начальной загрузки

Для добавления модуля integrity в initrd нужно выполнить команду:

```
sudo echo 'add_dracutmodules+=" integrity "' | tee /etc/dracut.conf.d/10-ima.conf
```

Затем нужно пересобрать образ начальной загрузки initrd командой:

```
sudo systemd-initramfs-gen
```

Следует обратить внимание, что в initrd должны попасть уже подписанные файлы, поэтому данный шаг выполняется после подписывания файлов в Системе.

7.4.7 Пробный запуск подписанной Системы

После подготовительных настроек ЗПС следует осуществить пробный запуск Системы с подписанными исполняемыми файлами. Для этого нужно добавить параметр загрузки ядра ima_appraise=log в начальный загрузчик

GRUB, аналогично описанному в п.3.8. В этом режиме Система не будет запрещать запуск неподписанных файлов, но будет записывать попытки их запуска.

После запуска Системы с вышеуказанным ключом можно попробовать проверить подпись системных файлов, например, интерпретатора bash, командой:

```
sudo ima_inspect /bin/bash
```

В выводе команды должна быть указана подпись и ее тип – streebog512, проверка подписи должна заканчиваться строчкой:

```
/bin/bash: verification is OK
```

7.4.8 Рабочий запуск и проверка работы

Для рабочего использования необходимо запустить Систему без параметра ima_appraise=log в начальный загрузчик GRUB, то есть просто перезагрузить Систему. Система должна запуститься и работать штатно.

Далее необходимо проверить работу защиты от запуска неподписанных исполняемых файлов:

а) Войти в root-консоль с помощью команды:

```
sudo -i
```

б) Создать копию существующего исполняемого файла:

```
cp -v /bin/cat cat.copy
```

в) Предоставить права на исполнение для файла:

```
chmod +x cat.copy
```

г) Попробовать запустить файл командой:

```
./cat.copy
```

д) Получить в выводе ошибку:

```
-bash: ./cat.copy: Отказано в доступе
```

При этом в журнале аудита должна появиться запись о запрете запуска неподписанного файла.

Следует удалить из Системы закрытый ключ после проверки подписи файлов и штатной работоспособности ОС.

7.5 Конфигурация системы для защиты информации

7.5.1 Общее описание

Для использования ОС в задачах, требующих повышенной защиты информации, применяются более строгие настройки файлов конфигурации Системы и более подробные настройки журналирования событий. ОС имеет возможность использования таких настроек благодаря установке пакетов `sconfigs`. При этом будут установлены пакеты, добавляющие в ОС настройки, обеспечивающие повышенную защиту информации и которые можно выбрать в зависимости от текущего профиля защиты.

ОС может быть сконфигурирована для безопасной работы используя настройки, описанные в пп.7.5.2-7.5.9.

7.5.2 Настройки службы SSH

Безопасные настройки службы удаленного доступа SSH – sshd – обеспечиваются установкой пакета sconfigs-sshd.

При этом устанавливаются ключи настроек в файле /etc/ssh/sshd_config.d/sshd_hardening.conf .

Все настройки в файле подробно документированы и при необходимости могут быть изменены опытным администратором. Более подробно настройка удаленного доступа к Системе через SSH описана в разделе 8.1.

7.5.3 Отключение файловых систем без атрибутов безопасности

Для безопасности рекомендуется отключать возможность загрузки редко используемых ФС, не поддерживающих атрибуты безопасности, с помощью команды `modprobe`.

Эти настройки могут быть выполнены с помощью установки пакета sconfigs-disable-filesystems-modules и связанных с ним служебных пакетов.

Документированный список отключаемых ФС настраивается в файле /etc/modprobe.d/restrict_fs_modules.conf.

7.5.4 Отключение редких сетевых протоколов

Отключение в целях безопасности неиспользуемых сетевых протоколов может быть реализовано с помощью установки пакета sconfigs-disable-network-modules с зависимыми пакетами. После его установки в Систему

добавляется файл настроек /etc/modprobe.d/restrict_net_modules.conf с документированными настройками отключения таких протоколов.

7.5.5 Отключение дампов оперативной памяти

ОС по умолчанию сохраняет дампы оперативной памяти (coredumps) при сбоях Системы, что позволяет искать и исправлять ошибки. Но такая настройка снижает безопасность Системы, т.к. в дампах могут содержаться открытые пароли и другие важные данные. Рекомендуется отключать возможность создания дампов памяти путем установки пакета sconfigs-disable-coredumps.

При установке пакета в файлах конфигурации /etc/security/limits.d/90-disable-coredumps.conf и /etc/sysctl.d/96-disable-coredumps.conf будут установлены необходимые для этого отключения настройки.

7.5.6 Настройка переменных ядра

Для повышения безопасности функционирования ядра используется множество настроек (например, отключение системных вызовов по клавише SysRq). Установить всю совокупность этих настроек можно установкой пакета sconfigs-harden-kernel. Эти настройки перечислены и документированы в файле /etc/sysctl.d.95-kernel-hardening.conf.

7.5.7 Настройки аудита

Необходимым условием защиты информации в ОС является настройка и функционирование службы аудита. В репозиториях предлагается два шаблона настроек аудита – сокращенный sconfigs-basic-audit и полный sconfigs-detailed-audit. Эти пакеты содержат, соответственно, файлы настроек аудита /etc/sconfigs/audit-rules/20-basic-audit.rules и /etc/sconfigs/audit-rules/30-detailed-audit.rules, в которых подробно документированы настройки службы аудита и установлены их рекомендованные значения.

7.5.8 Отключение настроек блокировки экрана

По умолчанию пользователь в KDE Plasma может отключить блокировку экрана по наступлении периода бездействия или при выходе из ждущего режима, что позволяет неавторизованному пользователю получить доступ к Системе. Файл /etc/xdg/kscreenlockerrc из пакета sconfigs-kscreenlocker блокирует такое поведение, делая настройки по умолчанию недоступными для изменения.

7.5.9 Настройка автоблокировки консольного режима

Для работы в консольном режиме tty применяется программа tmux, сконфигурированная (файл /etc/tmux.conf) следующим образом:

```
    set -g lock-command "vlock -c"
    set -g lock-after-time 300
    bind L lock-session
```

Таким образом включается автоблокировка консоли по истечении 5 минут бездействия командой `vlock`.

Замена стандартной консоли tty на tmux осуществляется в файле конфигурации /etc/bashrc с помощью скрипта:

```
    if          [          -n          "$PS1"          ]          &&
[        "$XDG_SESSION_TYPE"        =        tty        ]        &&
[             -z             "$TMUX"             ]             &&
!       [[       "$TERM"       =~       screen       ]]       &&
!       [[       "$TERM"       =~       tmux       ]]       &&
command -v tmux >/dev/null 2>&1
    then
exec tmux
    fi
```

7.6 Аудит

7.6.1 Общие сведения о auditd

Аудит (audit) – это механизм аудита безопасности в ядре Linux-подобных ОС, который позволяет отслеживать события в Системе. Аудит может использоваться для мониторинга доступа к файлам, сетевых подключений, а также других системных событий.

В пространстве пользователя работает сервис auditd, который получает от ядра ОС информацию о произошедших событиях и обрабатывает ее в соответствии с настройками.

Ведение логов аудита обычно решает две задачи:

– протоколирование событий для разбора в случае действий злоумышленника;

– протоколирование событий для отладки работы программ.

7.6.2 Управление службой аудита

Служба auditd по умолчанию включена, однако при необходимости ее можно включить в автозагрузку при старте вручную:

```
sudo systemctl enable auditd
```

Выключить службу можно с помощью команды:

```
sudo systemctl mask auditd
```

Остановить работу службы можно с помощью команды:

```
sudo systemctl kill auditd
```

7.6.3 Настройка конфигурации аудита

Конфигурационный файл находится по адресу /etc/audit/auditd.conf. Его содержимое по умолчанию подходит для работы в Системе.

Некоторые из наиболее важных конфигурационных параметров:

– max_log_file – максимальный размер журнального файла аудита в Мб;
– num_logs – максимальное количество журнальных файлов аудита;
– log_file – путь к журнальному файлу аудита;
– log_group – группа, которой принадлежит журнальный файл аудита;
– write_logs – записывать (опция yes) или не записывать (опция no) события аудита в файл /var/log/audit/audit.log.

Примечание – Если установлена опция no, то события аудита будут попадать в systemd-journald, но не будут дублироваться в еще один файл. Утилита `ausearch` читает именно этот файл, поэтому ей нельзя будет пользоваться.

Больше информации можно получить по справке командой:

```
man auditd.conf
```

7.6.4 Настройка правил аудита

Правила аудита — это инструкции для системы аудита, определяющие, какие события и действия пользователей или процессов должны подвергаться аудиту, т.е. логироваться. Они помогают отслеживать действия, связанные с безопасностью, и регистрировать важные изменения в Системе. Настройка правил аудита в ОС осуществляется с помощью утилиты `auditctl`, которая взаимодействует с сервисом auditd.

Правила задают условия и параметры для мониторинга определенных файлов, каталогов или системных вызовов. Эти правила могут быть установлены

в реальном времени или загружены при запуске Системы из конфигурационных файлов.

`auditctl` позволяет задать правила аудита в режиме реального времени. Эти правила действуют только до момента перезагрузки Системы. Чтобы обеспечить постоянную работу правил аудита при каждой загрузке ОС, их нужно сохранять в конфигурационных файлах.

Правила аудита хранятся в следующих файлах:

– /etc/audit/audit.rules;
– /etc/audit/rules.d/*.rules.

Файлы содержат инструкции, каждая строка которых представляет собой аргумент для команды `auditctl`. При запуске auditd из этих файлов загружаются все правила, обеспечивая автоматическое применение аудита без необходимости ручного ввода команд при каждом запуске Системы.

Пример строки в файле правил:

```
-w /etc/passwd -p wa -k passwd-file
```

Эта строка аналогична следующей команде, которая будет выполнена утилитой `auditctl`:

```
sudo auditctl -w /etc/passwd -p wa -k passwd-file
```

В данном примере параметры означают следующее:

– -w /etc/passwd – указывает путь к файлу, который нужно отслеживать;
– -p wa – устанавливает права мониторинга: w (изменение) и a (добавление);
– -k passwd-file – добавляет метку (ключ) для поиска соответствующих событий в логах.

Рассмотрим основные типы параметров в правилах аудита:

– -w (watch) – указывает на файл или каталог, который нужно отслеживать;
– -p (permissions) – задает права доступа, за изменениями которых нужно следить. Доступные значения:

  – r – чтение файла;
  – w – запись в файл;
  – x – выполнение файла;
  – a – добавление информации в файл.

– -k (key) – метка для фильтрации и поиска событий по ключу;
– -S (syscalls) – отслеживание системных вызовов (например, open, execve);

– -F (fields) – добавление дополнительных условий, таких как идентификатор пользователя (UID) или группы (GID);

– -a (action) – определяет действие: добавление или удаление правила.

Для добавления правила аудита через командную строку можно использовать `auditctl`. Например, чтобы следить за изменениями в файле /etc/shadow, используется следующая команда:

```
sudo auditctl -w /etc/shadow -p wa -k shadow-file
```

Это правило будет активно до перезагрузки Системы.

Для удаления правил аудита используется параметр -d. Пример:

```
sudo auditctl -d -w /etc/shadow
```

7.6.4.1 Использование ключа поиска событий

Метка, установленная через параметр -k, позволяет легко находить события, связанные с этим ключом, в логах. Чтобы просмотреть события, связанные с конкретным правилом, можно использовать команду:

```
ausearch -k shadow-file
```

Для получения подробной справки по утилите `ausearch`, включая ключи поиска и другие опции, используется следующая команда:

```
man ausearch
```

7.6.5 Выборка из логов аудита

Утилита `ausearch` – это инструмент для анализа журналов аудита, созданных с помощью системы аудита ОС. Журналы аудита содержат записи о действиях, совершаемых в Системе, таких как изменение файлов, запуск процессов или использование сетевых ресурсов. Утилита `ausearch` предназначена для поиска и анализа этих записей.

Утилита может выполнять поиск записей журнала аудита по различным критериям, таким как время, пользователь, тип события и др.

В качестве примера можно использовать следующую команду для поиска всех записей журнала аудита, связанных с изменением файла /etc/passwd:

```
sudo ausearch -f /etc/passwd
```

Результаты поиска будут выведены на экран консоли. Каждая запись журнала аудита будет содержать информацию о времени события, типе события, пользователе, который совершил действие, и другие подробности.

Одной из наиболее полезных функций утилиты `ausearch` является возможность использования ключей для выполнения расширенного поиска. Например, следующая команда выполняет поиск записей журнала аудита, связанных с запуском процесса с именем httpd:

```
sudo ausearch -c httpd
```

Другим полезным ключом является ключ -i, который позволяет просмотреть подробную информацию о конкретной записи журнала аудита. Например, следующая команда выводит подробную информацию о записи журнала аудита:

```
sudo ausearch -i 12345
```

где 12345 – это идентификатор записи журнала аудита.

Утилита `ausearch` также может использоваться для выполнения аудита безопасности. Например, можно настроить систему аудита таким образом, чтобы она записывала информацию обо всех неудачных попытках входа в Систему. Кроме того, можно использовать утилиту `ausearch` для поиска этих записей журнала аудита и выявления потенциальных угроз безопасности.

Также утилита `ausearch` поддерживает несколько форматов вывода, включая форматы текста и JSON. Это может быть полезно для автоматизации процесса анализа журналов аудита.

7.6.6 Примеры работы со службой аудита

7.6.6.1 Отслеживание изменений в директории или файле

Для отслеживания всех изменений в директории /var/log нужно добавить следующее правило:

```
sudo auditctl -w /var/log -p w -k var_log_changes
```

Это правило отслеживает любые операции записи (опция w) в директорию /var/log и связывает их с ключевым словом (tag) var_log_changes.

Следует проверить, что правила аудита корректно добавлены командой, которая выводит список текущих правил аудита:

```
sudo auditctl -l
```

Для примера можно записать текст в файл внутри тестовой директории:

```
echo text | sudo tee /var/log/testfile
```

Можно просмотреть журнальные файлы аудита, используя утилиту ausearch. Чтобы просмотреть все записи аудита, связанные с директорией /var/log, следует использовать следующую команду:

```
sudo ausearch -k var_log_changes
```

Эта команда выводит все записи аудита, связанные с ключевым словом var_log_changes, в т.ч. событие записи в файл /var/log/testfile.

7.6.6.2 Отслеживание запуска определенного приложения

В данном случае используется правило аудита следующего вида:

```
sudo auditctl -a exit,always -F path=/usr/bin/myapp -F perm=x
-k myapp_execution
```

Это правило отслеживает каждый выход (exit) из приложения /usr/bin/myapp с правами на выполнение (x) и связывает его с ключевым словом myapp_execution.

Поиск событий осуществляется по аналогии с предыдущим пунктом:

```
sudo ausearch -k myapp_execution
```

7.6.6.3 Отслеживание системных вызовов

Правило аудита для отслеживания 64-битных системных вызовов execve() от имени root, т.е. отслеживания запуска программ от имени root:

```
sudo auditctl -a exit,always -F arch=b64 -S execve -F uid=0
-k authentication_events
```

Это правило отслеживает каждый вызов функции execve (используется для запуска исполняемых файлов) с идентификатором пользователя (uid) равным 0 (root) и связывает его с ключевым словом authentication_events. Так как 64-битные ОС поддерживают запуск и 32-битных программ, то можно подвергнуть аудиту то же самое в 32-битном варианте:

```
sudo auditctl -a exit,always -F arch=b32 -S execve -F uid=0
-k authentication_events
```

7.6.6.4 Отслеживание доступа к конфиденциальным файлам

Для отслеживания доступа к конфиденциальным файлам используется правило аудита следующего вида:

```
sudo auditctl -a exit,always -F arch=b64 -S open -F
dir=/etc/sensitive -F success=0 -k sensitive_files_access
```

Это правило отслеживает каждый вызов функции open для любого файла в директории /etc/sensitive, который заканчивается неудачей (success=0) и связывает его с ключевым словом sensitive_files_access.

7.6.6.5 Отслеживание сетевых подключений

Для того, чтобы настроить правила аудита на отслеживание всех сетевых подключений в Системе используется правило следующего вида:

```
    sudo auditctl -a exit,always -F arch=b64 -S bind -S connect
-F success=0 -k network_events
```

Это правило отслеживает каждый вызов функций bind() и connect() для любого сетевого подключения, которое заканчивается неудачей (success=0) и связывает его с ключевым словом network_events.

7.6.6.6 Запись в журнал аудита при подключении устройства USB

Для того, чтобы ОС фиксировала такие события как подключение USB-устройств нужно использовать следующее правило аудита следующего вида:

```
    sudo auditctl -w /dev/bus/usb -p rwxa -k usb
```

Теперь Система будет записывать информацию при подключении USB-устройства в журнал аудита.

Чтобы найти записи журнала аудита, связанные с подключением USB-устройства, нужно выполнить следующую команду ausearch:

```
    sudo ausearch -f /dev/bus/usb -k usb -i
```

Эта команда найдет все записи журнала аудита, связанные с ключом usb и событием OPEN, которое относится к подключению USB-устройства.

7.6.6.7 Централизованный сбор событий аудита

Централизованный сбор данных аудита в ОС необходим для долгосрочного хранения и анализа данных на внешних системах. Такой подход позволяет избежать потерь данных при повреждении локального диска и предоставляет возможность анализа журналов в централизованной точке. В данном контексте важны две основные задачи: архивирование событий и безопасная передача данных.

Для сохранения локальных журналов перед их отправкой на внешнее хранилище может использоваться команда `tar`, которая позволяет собрать все необходимые файлы с журналами событий в один архив, что упрощает передачу

и сжатие данных. Например, чтобы архивировать все журналы аудита, можно использовать команду:

```
tar -czvf audit_logs.tar.gz /var/log/audit
```

где параметры:

– -c – создает архив;
– -z – сжимает файлы с помощью gzip;
– -v – вывести процесс на экран;
– -f – указать имя файла.

Это архивирование важно для сохранения целостности данных до их передачи на внешние серверы.

Для отправки архивированных журналов событий на внешние хранилища используется служба systemd-journal-gatewayd. Эта служба предоставляет веб-интерфейс для просмотра и передачи логов по HTTP(S). Она может быть настроена для предоставления доступа как к локальным журналам, так и для передачи данных на внешние серверы.

Настройка этой службы выполняется через два файла конфигурации:

– /etc/systemd/journal-remote.conf – файл конфигурации, отвечающий за прием данных из внешних источников;
– /etc/systemd/journal-upload.conf – файл, используемый для настройки отправки данных на удаленные сервера.

Пример настройки конфигурации: в файле journal-upload.conf можно указать параметры для передачи данных на удаленный сервер:

```
[Upload]
URL=https://external-server:19532
```

Для обеспечения защищенной передачи данных аудита может использоваться шифрование с помощью SSL/TLS. Средство OpenSSL позволяет создавать ключи для защиты передаваемых данных. Перед передачей на сервере генерируются ключи и сертификаты:

а) Создание приватного ключа:

```
openssl genrsa -out private.key 2048
```

б) Генерация сертификата:

```
openssl req -new -x509 -key private.key -out server.crt -days 365
```

Эти ключи используются в конфигурации journal-upload.conf для обеспечения шифрования.

Эти меры безопасности позволяют гарантировать, что данные аудита будут переданы в защищенном виде и не будут перехвачены злоумышленниками.

7.7 Отказоустойчивый кластер

7.7.1 Общие сведения

Pacemaker и Corosync являются компонентами отказоустойчивой кластерной среды с открытым исходным кодом, предназначенной для обеспечения высокой доступности сервисов.

Pacemaker представляет собой менеджер ресурсов кластера (Cluster Resource Manager, CRM), отвечающий за управление состоянием ресурсов в кластере и их автоматическое переключение при отказах. Corosync обеспечивает кластерное членство и обмен сообщениями между узлами, выступая в роли коммуникационного уровня.

Кластерная архитектура Pacemaker и Corosync условно делится на три уровня:

– CIB (Cluster Information Base) – уровень, независимый от кластера. Содержит конфигурацию ресурсов и агентов управления. CIB представляет собой XML-базу, реплицируемую между всеми узлами кластера. Управление осуществляется с любого узла;

– Cluster Resource Manager (CRMd) – уровень управления ресурсами, реализуемый демоном pacemaker. Отвечает за реакцию на события, происходящие в кластере (отказы, переключения узлов, действия администратора), расчет оптимального состояния и реализацию необходимых операций;

– PEngine (Policy Engine) – информационный уровень. Обеспечивает сетевое взаимодействие, обработку кворума, распределение команд управления и реализацию политик работы кластера. Этот уровень опирается на Corosync.

Ключевые компоненты кластера:

– DC (Designated Coordinator) – основной управляющий узел, выбранный путём голосования. При его недоступности выполняется автоматическая переизбрание;

– LRMd (Local Resource Manager daemon) – демон управления локальными ресурсами. Отвечает за исполнение команд через RA (Resource Agent) – скрипты, управляющие жизненным циклом ресурса;

– STONITHd (Shoot The Other Node In The Head) – компонент защиты от порчи данных. Обеспечивает физическое отключение узла в случае критического сбоя, в том числе через IPMI, BMC или внешние устройства.

С точки зрения Pacemaker ресурсом считается любой объект, управляемый через скрипты: от IP-адреса и службы Apache до систем с репликацией и режимами master/slave. Управление ресурсами должно осуществляться исключительно через кластер, исключая ручной запуск или автоматический запуск при загрузке ОС.

7.7.1.1 Типы ресурсов кластера

В Pacemaker реализованы следующие типы ресурсов:

– Primitive – базовый тип, представляющий собой один экземпляр ресурса;

– Clone – ресурс, который может быть запущен на нескольких узлах. Делится на типы: Anonymous, Globally Unique и Stateful;

– Multi-state – расширение Clone-ресурса с поддержкой состояний Master и Slave. Применяется, например, для DRBD.

Дополнительно ресурсы могут объединяться в Group, которая обеспечивает запуск на одном узле в заданном порядке.

Поддерживаются механизмы:

– Migration – "живой" перенос ресурса между узлами (при поддержке агентом операций migrate_to и migrate_from);

– Colocation – указание на необходимость совместного или раздельного размещения ресурсов;

– Ordering – определение порядка запуска и остановки ресурсов;

– Location – указание предпочтений по размещению ресурсов на узлах.

7.7.1.2 Политики управления кластера

Pacemaker позволяет управлять поведением ресурсов при отказах и восстановлении узлов с помощью параметров:

– resource-stickiness – желательность оставаться на текущем узле;

– migration-threshold – количество отказов, после которых ресурс перемещается;

– failure-timeout – интервал, после которого считается, что сбой более неактуален;

– multiple-active – поведение при запуске ресурса на нескольких узлах;

– monitor – периодическая проверка состояния ресурса (указывается через параметр interval).

Уровень кворума определяет возможность функционирования кластера. По умолчанию кластер считается работоспособным при наличии более 50%

работоспособных узлов. Поведение при потере кворума задаётся параметром no-quorum-policy.

Pacemaker поддерживает разнообразные схемы отказоустойчивости: Active/Passive, Active/Active, N+1, N-to-1 и другие.

7.7.2 Развёртывание кластера высокой доступности

Данный раздел содержит пошаговую инструкцию по развёртыванию кластера высокой доступности на двух серверах под управлением ОС РОСА "ХРОМ" с использованием компонентов Pacemaker и Corosync. В качестве примера рассматривается настройка кластера с веб-сервером Nginx:

а) Подготовка Системы.

На всех узлах, планируемых к объединению в кластер, необходимо внести записи в файл /etc/hosts, чтобы обеспечить корректное разрешение имён.

Пример содержимого файла /etc/hosts:

```
10.45.4.58 webserver-01
10.45.4.59 webserver-02
```

Редактирование может быть выполнено с использованием любого доступного текстового редактора.

б) Установка и первичная настройка Nginx.

На каждом сервере устанавливается веб-сервер Nginx с помощью менеджера пакетов. После установки служба добавляется в автозагрузку и запускается командами:

```
sudo dnf install nginx
sudo systemctl enable nginx
sudo systemctl start nginx
```

Для визуального различения узлов в кластере рекомендуется изменить содержимое страницы по умолчанию. На каждом сервере в файл /usr/share/nginx/html/index.html добавляется уникальная метка.

На сервере webserver-01:

```
echo "<h1>webserver-01</h1>" >\
/usr/share/nginx/html/index.html
```

На сервере webserver-02:

```
echo "<h1>webserver-02</h1>" >\
/usr/share/nginx/html/index.html
```

в) Установка компонентов Pacemaker и Corosync.

Для настройки кластера устанавливаются пакеты corosync, pacemaker и pcsd. Указанные службы добавляются в автозагрузку и запускаются.

```
sudo dnf install corosync pacemaker
sudo systemctl enable corosync
sudo systemctl enable pacemaker
sudo systemctl enable pcsd
sudo systemctl start corosync
sudo systemctl start pacemaker
sudo systemctl start pcsd
```

В процессе установки создаётся системный пользователь hacluster, необходимый для аутентификации между узлами. На всех узлах кластера для пользователя hacluster задаётся одинаковый пароль.

```
passwd hacluster
```

г) Аутентификация и создание кластера.

После установки и настройки компонентов необходимо выполнить аутентификацию между узлами с использованием утилиты pcs:

```
sudo pcs cluster auth webserver-01 webserver-02
```

Затем создаётся кластер с именем rosacluster, включающий оба сервера:

```
sudo pcs cluster setup --force --name rosacluster webserver-01 webserver-02
```

Для корректного функционирования кластера рекомендуется повторно включить службы в автозагрузку и перезагрузить Систему:

```
sudo systemctl enable corosync
sudo systemctl enable pacemaker
sudo systemctl enable pcsd
reboot
```

д) Проверка состояния кластера после перезагрузки с помощью команды:

```
pcs status
```

7.7.3 Управление ресурсами и политиками в кластере

7.7.3.1 Отключение STONITH

STONITH (Shoot-The-Other-Node-In-The-Head) представляет собой механизм аппаратного ограждения, применяемый для исключения повреждения общих ресурсов в случае отказа узла. В производственной среде рекомендуется

включать STONITH и использовать совместимые устройства управления питанием или IPMI. Однако в демонстрационных или тестовых средах при отсутствии таких устройств функцию STONITH допускается отключить.

При отсутствии настроенных устройств STONITH при выполнении команды `pcs status` в выводе будет отображаться предупреждение.

Для отключения STONITH в кластере выполняется команда:

```
sudo pcs property set stonith-enabled=false
```

7.7.3.2 Отключение контроля кворума

По умолчанию кластер прекращает управление ресурсами, если теряет кворум (менее половины активных узлов). Для тестовой конфигурации, в частности двухузлового кластера, может быть необходимо отключить реакцию на потерю кворума. Это осуществляется с помощью настройки no-quorum-policy.

Отключение реакции на потерю кворума выполняется следующей командой:

```
sudo pcs property set no-quorum-policy=ignore
```

Текущие свойства кластера можно просмотреть командой:

```
pcs property list
```

7.7.3.3 Добавление ресурсов в кластер

Одним из ключевых элементов высокой доступности является плавающий IP-адрес – адрес, который может автоматически перемещаться между узлами в пределах одной сети. В приведённом примере используется IP-адрес 10.45.4.60.

Для регистрации IP-адреса в виде ресурса с именем v_ip используется следующая команда:

```
sudo pcs resource create v_ip ocf:heartbeat:IPaddr2
ip=10.45.4.60 cidr_netmask=24 op monitor interval=20s
```

Далее добавляется ресурс nginx, представляющий службу веб-сервера, управляемую через systemd:

```
sudo pcs resource create nginx systemd:nginx
```

Состояние ресурсов проверяется с помощью команды:

```
sudo pcs status resources
```

Если в выводе отображаются ресурсы v_ip и nginx, это означает, что плавающий IP-адрес и веб-сервер были успешно добавлены в кластер.

7.7.3.4 Настройка политики совместного размещения ресурсов

Для обеспечения корректной работы веб-сервера Nginx необходимо, чтобы он всегда запускался на том же узле, где активен плавающий IP-адрес. Для этого используется политика коллокации.

Настройка коллокации выполняется командой:

```
sudo pcs constraint colocation add nginx v_ip INFINITY
```

Также необходимо определить порядок запуска ресурсов: сначала должен быть активирован IP-адрес, затем – веб-сервер.

Настройка порядка запуска ресурсов:

```
sudo pcs constraint order v_ip then nginx
```

Проверка текущего состояния ресурсов кластера выполняется командой:

```
sudo pcs status
```

7.7.4 Проверка работоспособности кластера

Для проверки корректности настройки отказоустойчивого кластера необходимо убедиться, что ресурсы (веб-сервер Nginx и плавающий IP-адрес) автоматически переключаются между узлами в случае отказа одного из них.

Для выполнения проверки необходимо:

а) Открыть веб-браузер и перейти по адресу:

```
http://10.45.4.60
```

На открывшейся странице можно увидеть индексную страницу Nginx, размещённую на первом узле webserver-01.

б) Для эмуляции отказа текущего ведущего узла выполняется остановка кластера на webserver-01:

```
sudo pcs cluster stop webserver-01
```

в) После остановки кластера на первом узле при повторном обращении к адресу http://10.45.4.60 будет отображена индексная страница, размещённая на узле webserver-02. Это означает, что ресурсы кластера автоматически переместились на доступный узел, обеспечив непрерывность сервиса.

Проверка выполнения всех шагов позволяет подтвердить успешную настройку отказоустойчивого кластера с использованием Pacemaker и Corosync.

7.7.5 Мониторинг и управление отказами

Для обеспечения корректной работы ресурсов в кластере необходимо настроить их регулярную проверку. В Pacemaker для этого используется операция monitor.

Операция мониторинга добавляется в момент создания ресурса с указанием интервала:

```
sudo pcs resource create nginx systemd:nginx op monitor interval=30s
```

Если ресурс уже создан, параметр можно добавить отдельно:

```
sudo pcs resource op add nginx monitor interval=30s
```

Pacemaker отслеживает результат работы проверки и реагирует согласно установленной политике. Рекомендуется также задать параметры, определяющие поведение кластера при сбоях:

– migration-threshold – количество сбоев, после которого ресурс переносится на другой узел;

– failure-timeout – интервал (в секундах), через который сбой считается неактуальным и может быть повторно допущен;

– on-fail – политика действия при сбое (restart, fence, standby, block и др.).

Пример команды:

```
sudo pcs resource update nginx migration-threshold=2 failure-timeout=60 on-fail=restart
```

Для отслеживания статуса ресурсов рекомендуется использовать команду:

```
sudo pcs status resources
```

Логирование событий выполняется службой Pacemaker и доступно через системный журнал:

```
journalctl -u pacemaker
```

7.7.6 Управление ресурсами и диагностика

В процессе эксплуатации кластера может потребоваться ручное вмешательство: перемещение ресурсов между узлами, временное отключение ресурса, снятие ограничений.

Перемещение ресурса на конкретный узел:

```
sudo pcs resource move nginx webserver-02
```

Запрет запуска ресурса на узле:

```
sudo pcs resource ban nginx webserver-01
```

Очистка всех ограничений, связанных с ресурсом:

```
sudo pcs resource clear nginx
```

Временное отключение ресурса:

```
sudo pcs resource disable nginx
```

Повторное включение ресурса:

```
sudo pcs resource enable nginx
```

Просмотр состояния кластера с детализацией:

```
sudo pcs status
sudo pcs status resources
sudo pcs status nodes
```

Дополнительно может использоваться команда для получения краткого отчёта по текущему состоянию кластера:

```
sudo crm_mon -1
```

Рекомендуется регулярно сохранять конфигурацию кластера:

```
sudo pcs config backup rosacluster-backup.tar.gz
```

Для восстановления используется команда:

```
sudo pcs config restore rosacluster-backup.tar.gz
```

8 УДАЛЕННЫЙ ДОСТУП К СЕРВЕРУ

Часть из рассматриваемых далее способов удаленного подключения к Системе работает только в случае, если установлен GUI-интерфейс.

8.1 Доступ по OpenSSH

8.1.1 Общие сведения об OpenSSH

OpenSSH – это бесплатный SSH-сервер, дающий возможность интерактивного управления сервером.

В большинстве дистрибутивов OpenSSH-сервер уже присутствует в ОС, и его установка не требуется. В случае отсутствия OpenSSH следующая команда выполнит установку необходимых пакетов:

```
sudo dnf openssh
```

Далее следует добавить SSH-сервер в автозагрузку. При следующем запуске сервера ОС выполнит автоматический запуск SSH-сервера. Как и в случае с другими сервисами, systemd позволяет управлять параметрами запуска, автозагрузки и рестарта демона OpenSSH. Автозапуск включается командой:

```
sudo systemctl enable ssh
```

Работоспособность утилиты проверяется командой:

```
ssh localhost
```

8.1.2 Настройка SSH

Настройка SSH необходима для улучшения защищенности Системы. Например, можно отключить возможность входа от имени пользователя root или изменить порт подключения со стандартного 22 на произвольный. Лучше использовать порты из верхнего диапазона (50000-65000).

Примечание – В стеке протоколов TCP/IP доступно 65536 портов.

Настройка выполняется в конфигурационном файле. Перед его модификацией создают резервную копию.

```
# sudo cp /etc/ssh/sshd_config
/etc/ssh/sshd_config.factory-defaults
```

Все изменения конфигурации SSH проводятся в файле /etc/ssh/sshd_config. Для его редактирования нужно выполнить команду:

```
sudo nano /etc/ssh/sshd_config
```

В файле нужно раскомментировать строку Port 22 и изменить значение на 55555 (рисунок 114).

Следует учитывать, что автоматизированные системы, такие как боты, зачастую приоритетно сканируют порты с одинаковыми цифрами. Поэтому в производственных и корпоративных сетях рекомендуется применять порты с различными цифрами для повышения уровня безопасности.

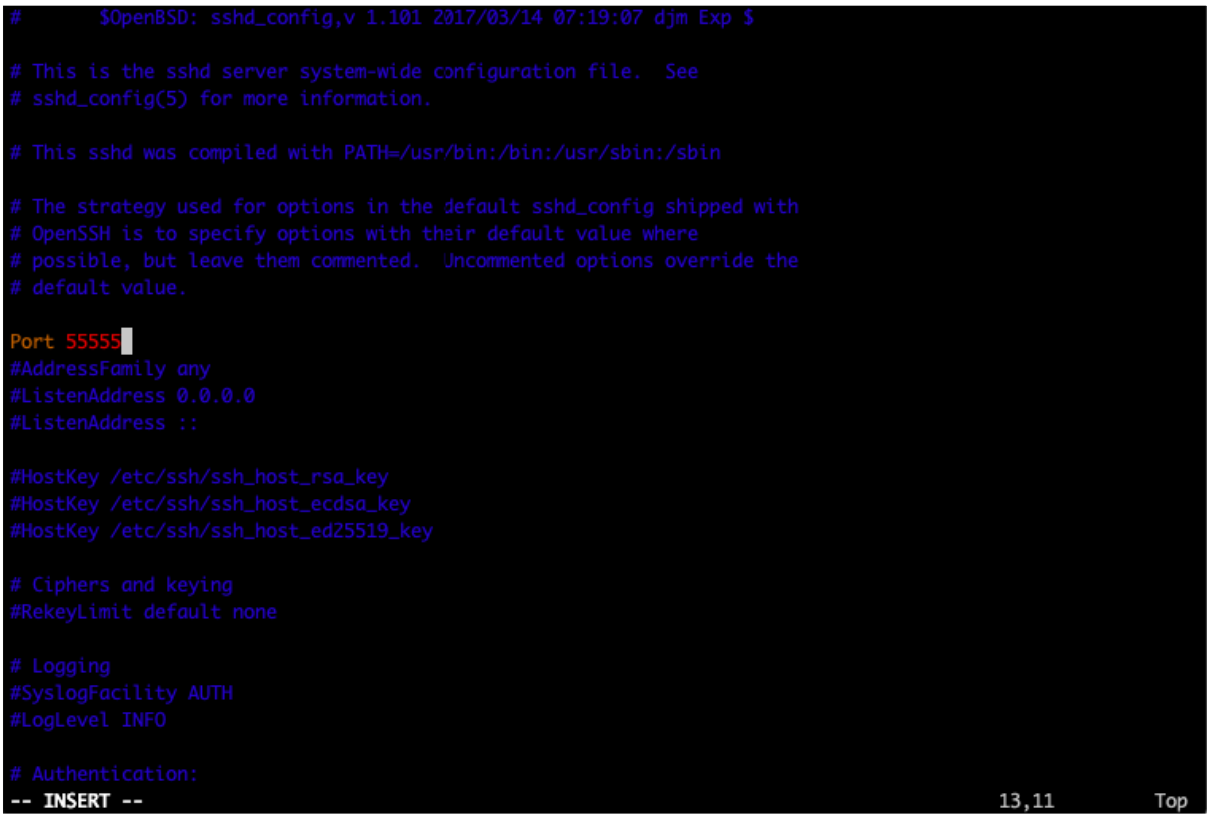


Рисунок 114 – Изменение стандартного порта подключения

Далее нужно отключить возможность входа на сервер учетной записи суперпользователя (root) и добавить возможность входа через ключи. Для этого необходимо изменить значения параметров PermitRootLogin на "no" и PubkeyAuthentication на "yes" (рисунок 115).



Рисунок 115 – Изменение параметров PermitRootLogin и PubkeyAuthentication

После этого следует перезагрузить демон SSH. Соединение при этом будет разорвано и подключиться можно будет через новый порт и пользовательскую учетную запись (она должны быть предварительно создана):

```
sudo systemctl restart ssh
```

Далее нужно переподключиться от обычной учетной записи и по другому порту:

```
ssh -p 55555 ansible@192.168.0.104
```

После успешного подключения можно продолжать работу с сервером.

8.1.3 Пары ключей

Еще один способ аутентификации на сервере – пара ключей RSA: открытый и закрытый. Открытый хранится на сервере, к которому будет выполняться подключение, а закрытый – на удаленном компьютере (или другом сервере), с которого выполняется подключение.

Далее (рисунок 116) в качестве примера приведена схема безопасного обмена ключами между "Отправителем" и "Получателем". "Злоумышленник" может читать сообщения, если они не зашифрованы. Здесь "Отправитель" или "Получатель" шифруют сообщение при помощи открытого ключа принимающей стороны, которая его дешифрует при помощи своего закрытого ключа.

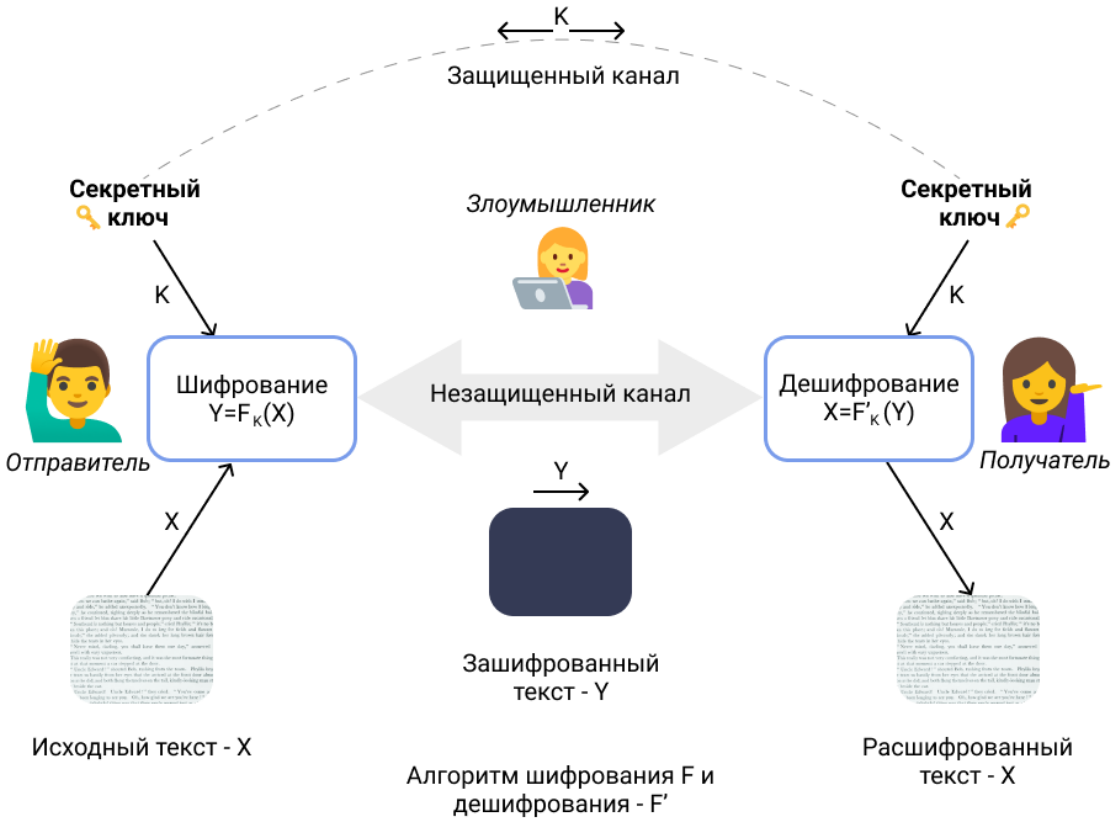


Рисунок 116 – Безопасный обмен ключами

Чтобы сгенерировать такую пару ключей, достаточно выполнить команду:

```
ssh-keygen -t rsa
```

Команду нужно выполнять на рабочей станции от имени пользователя, который будет в дальнейшем подключаться к удаленному компьютеру. Путь к хранению ключей можно оставить по умолчанию.

После создания ключей можно переходить к следующему шагу – копированию открытого ключа на удаленный сервер, выполнив следующую команду:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub rosa@192.168.0.104
```

В этом примере 192.168.0.104 – это IP-адрес удаленного сервера. После ввода пароля ключ копируется в папку .ssh домашней директории пользователя.

Сразу же после выполнения копирования следует проверить доступ при помощи созданной пары ключей:

```
ssh rosa@192.168.0.104
```

Для отключения возможности входа по паролю необходимо в файле /etc/ssh/sshd_config отредактировать значение PasswordAuthentication и присвоить "no" (рисунок 117).

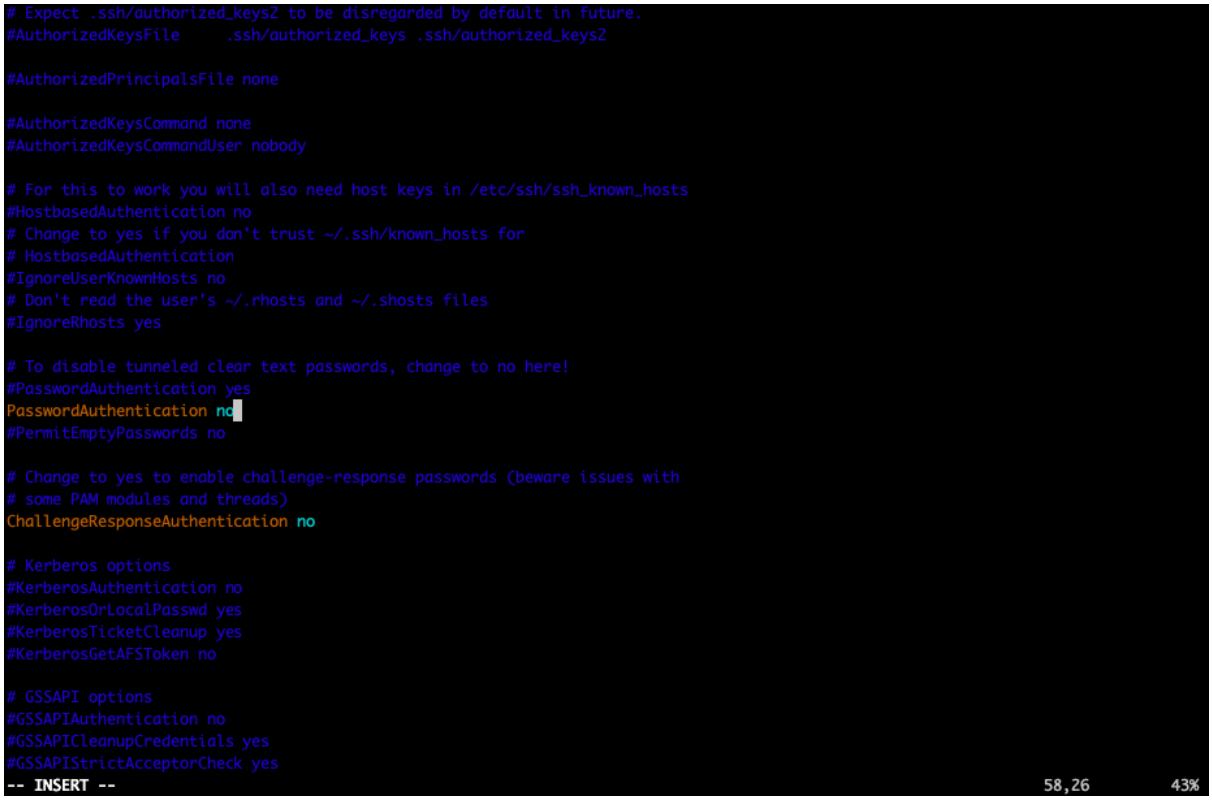


Рисунок 117 – Редактирование параметра PasswordAuthentication

После изменения настроек нужно перезагрузить службу SSH:

```
sudo systemctl restart ssh
```

После проведенных настроек при попытке подключения пользователем, для которого не определена пара ключей, будет выдаваться ошибка подключения.

При этом подключение при помощи ключа будет успешным.

Отключение доступа по паролю значительно повышает безопасность сервера.

8.1.4 Процессы, запущенные пользователями, сеанс которых завершен

При работе с сервером пользователь может запускать различные процессы (например, tmux или screen), завершение которых происходит по окончании сеанса пользователя.

После выхода пользователя из SSH-сессии Система автоматически завершает процессы, но в некоторых случаях выполнение определённых процессов необходимо сохранить.

Важно – Для выполнения дальнейших действий пользователю необходимо иметь несколько машин, подключение к которым происходит с помощью SSH. Настройка SSH описана в п. 8.1.2.

Для завершения процесса после выхода пользователя из SSH-сессии нужно выполнить следующие действия:

а) создать нового пользователя test1 на основной машине (рисунок 118);

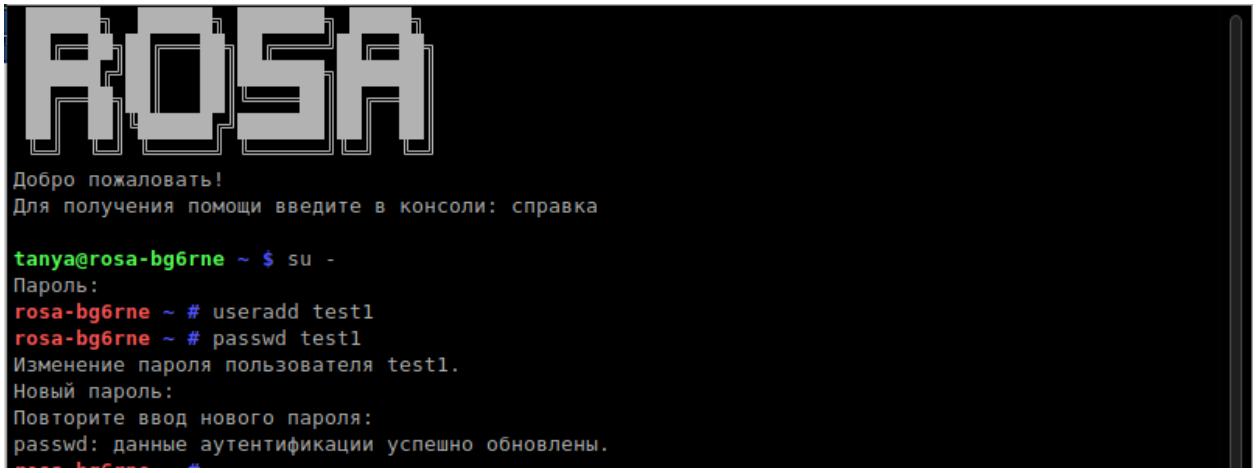


Рисунок 118 – Создание пользователя test1

б) подключиться к другой машине под именем нового пользователя с помощью следующей команды (рисунок 119):

```
ssh test1@192.168.1.131
```

где:

– test1 – имя пользователя;
– 192.168.1.131 – IP-адрес машины, к которой подключается пользователь.

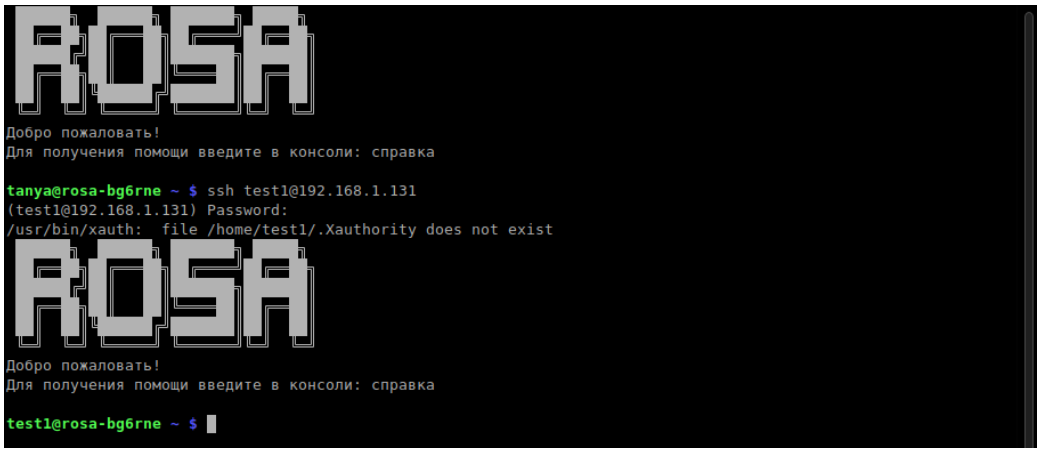


Рисунок 119 – Успешное подключение пользователя test1

в) выполнить команду screen, после чего должна открыться консоль;

г) в открывшейся консоли нужно выполнить команду `sleep infinity` (рисунок 120);

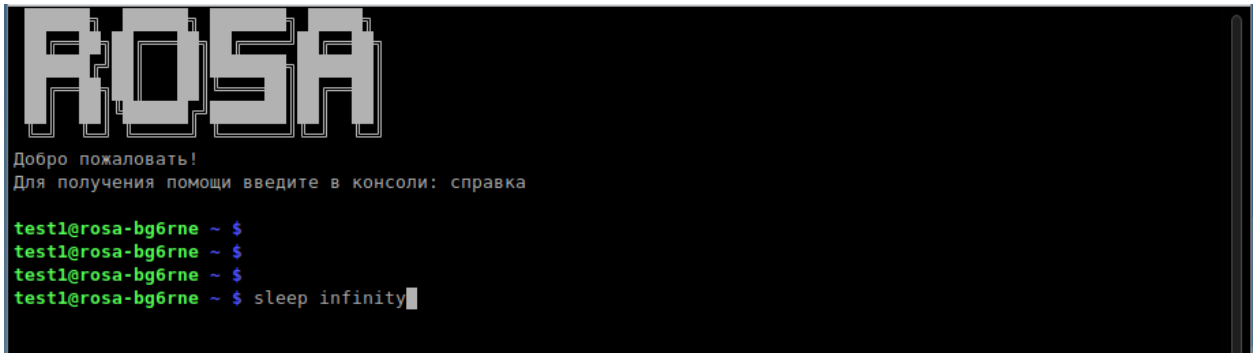


Рисунок 120 – Запуск процесса бесконечного ожидания

д) выйти из screen с помощью комбинации клавиш `Ctrl+A`, а затем `Ctrl+D`;

е) выйти из SSH-сессии одним из следующих способов:

– выполнить команду `exit`;
– использовать комбинацию клавиш `Ctrl+D`;
– закрыть консоль.

ж) открыть новую SSH-сессию с помощью команды `ssh test1@192.168.122.76` (рисунок 121);

з) вывести список запущенных сессий внутри screen с помощью команды `screen -r` (рисунок 121).

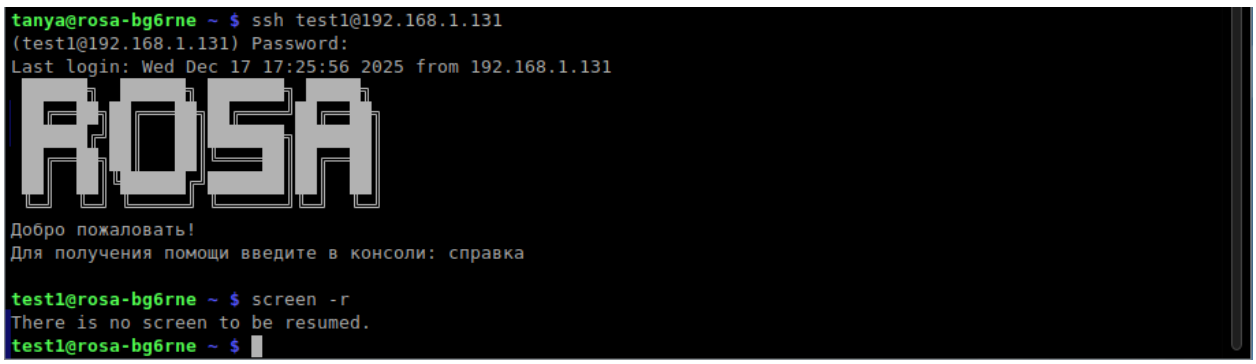


Рисунок 121 – Результат выполнения команды screen -r после повторного открытия SSH-сессии

В результате выполнения команды `screen -r` видно, что процессы screen и sleep были автоматически завершены после выхода пользователя из SSH-сессии.

При отсутствии активных screen-сеансов команда может вывести сообщение:

```
There is no screen to be resumed.
```

Для сохранения процессов после выхода пользователя из SSH-сессии можно воспользоваться одним из следующих вариантов:

– использовать команду `systemd-run --scope --user screen`;
– установить пакет systemd-settings-disable-kill-user-processes.

8.1.4.1 Использование команды сохранения определенного процесса

При использовании команды `systemd-run --scope --user screen` вместо screen указывается команда, запускающая определенный процесс.

В качестве примера использования команды `systemd-run --scope --user screen` можно выполнить следующие шаги:

а) начать SSH-сессию с помощью команды:

```
ssh test1@192.168.122.76
```

где test1 – имя пользователя, созданного в шаге а) п.8.1.4.

б) выполнить команду `systemd-run --scope --user screen` для запуска процесса screen;

в) выполнить команду `sleep 9999999h`, запускающую процесс ожидания (рисунок 122);

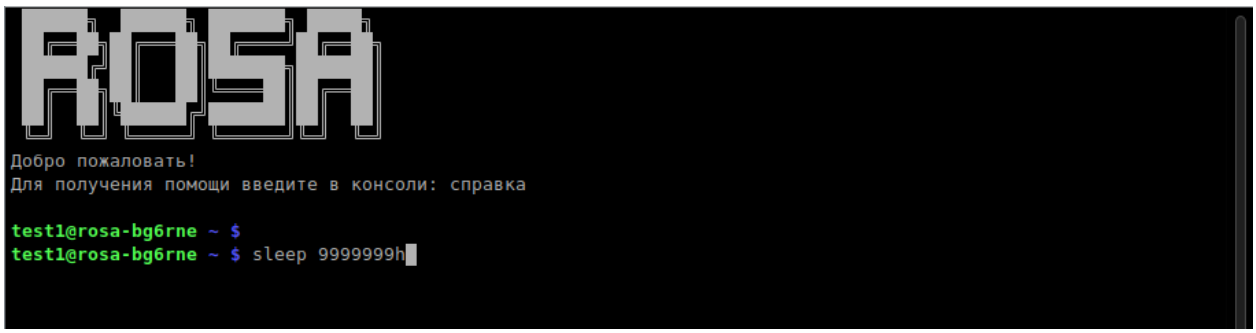


Рисунок 122 – Новый процесс ожидания

г) выйти из screen с помощью последовательного нажатия комбинаций клавиш `Ctrl+A` и `Ctrl+D`;

д) выйти из SSH-сессии одним из следующих способов:

– выполнить команду `exit`;

– использовать комбинацию клавиш `Ctrl+D`;

– закрыть консоль.

е) открыть новую SSH-сессию, как в шаге а), и вывести список запущенных сессий внутри screen с помощью команды `screen -r` (рисунок 123).

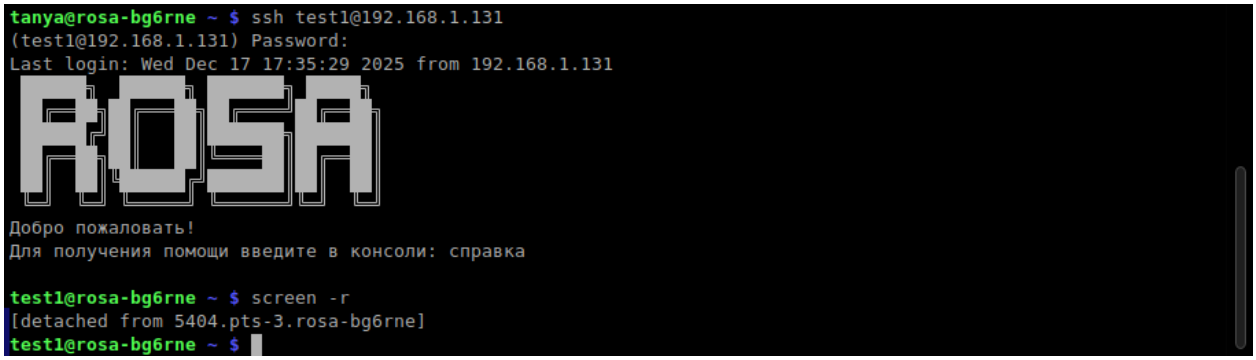


Рисунок 123 – Список запущенных процессов

В результате выполнения команды `screen -r` видно, что процессы screen и sleep, запущенные пользователем до выхода из SSH-сессии, не завершились после выхода из SSH-сессии.

Таким образом, с помощью команды `systemd-run --scope --user screen` можно сохранить начатый процесс и после закрытия пользователем SSH-сессии.

8.1.4.2 Пакет для сохранения пользовательских процессов после завершения SSH-сессии

Пакет systemd-settings-disable-kill-user-processes предотвращает уничтожение всех пользовательских процессов после завершения SSH-сессии, в которой они были запущены.

Важно – Установку пакета нужно выполнять с правами суперпользователя.

Для установки пакета systemd-settings-disable-kill-user-processes пользователю необходимо выполнить следующую команду (рисунок 124):

```
dnf install systemd-settings-disable-kill-user-processes
```

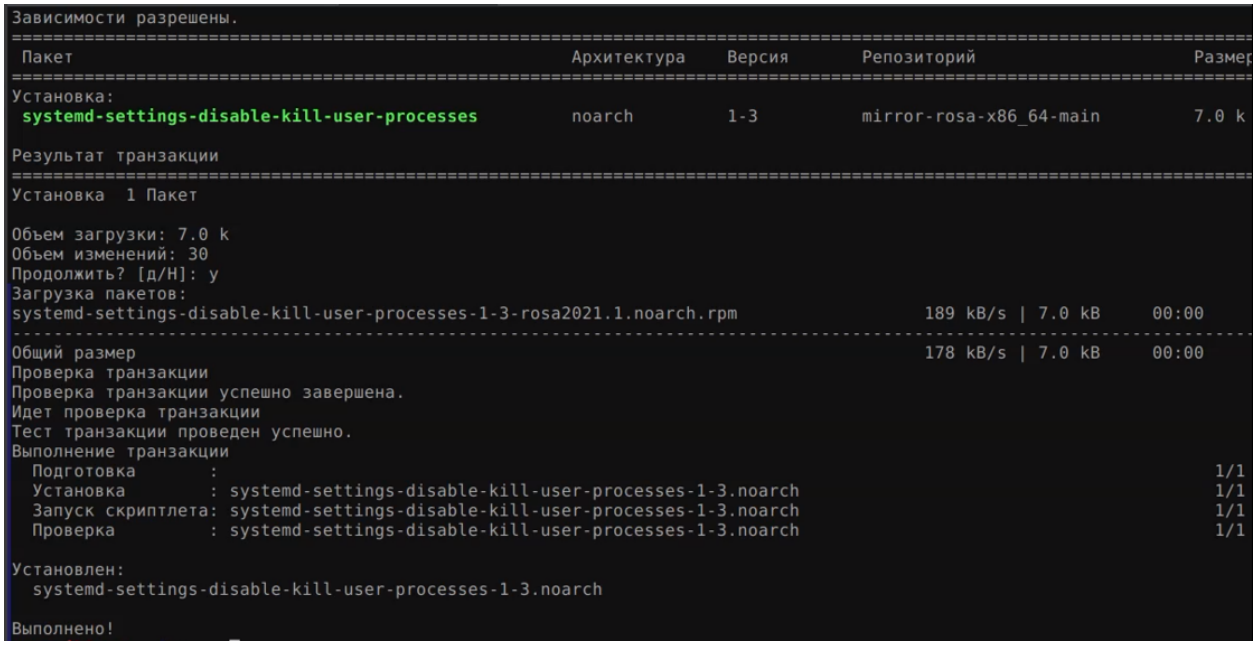


Рисунок 124 – Успешная установка пакета

Для применения настроек после установки пакета пользователю нужно выполнить перезагрузку Системы. Для перезапуска можно использовать одну из следующих команд:

– `reboot`;
– `systemctl reboot`.

Примечание – Пользователь может применить настройки без перезагрузки Системы с помощью команды:

```
systemctl restart systemd-logind.
```

Перезапуск systemd-logind завершит все активные пользовательские сессии.

8.1.4.3 Проверка сохранения процессов после установки пакета

После установки пакета systemd-settings-disable-kill-user-processes все пользовательские процессы должны продолжить выполнение по окончании сеанса пользователя.

Для проверки сохранения процессов нужно выполнить следующие шаги:

а) открыть SSH-сессию под именем нового пользователя с помощью следующей команды (рисунок 125):

```
ssh test1@192.168.1.131
```

где:

– test1 – имя пользователя, созданного в шаге а) п.8.1.4;

– 192.168.1.131 – IP-адрес машины, к которой подключается пользователь.

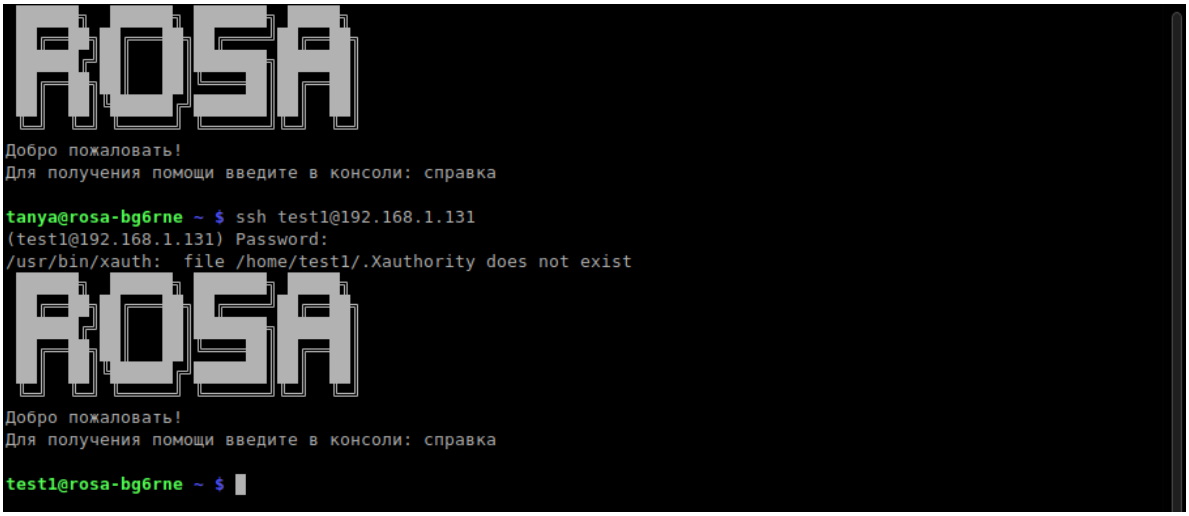


Рисунок 125 – Успешное подключение пользователя test1

б) выполнить команду `screen`, после чего должна открыться консоль;

в) в открывшейся консоли нужно выполнить команду `sleep infinity` (рисунок 126);

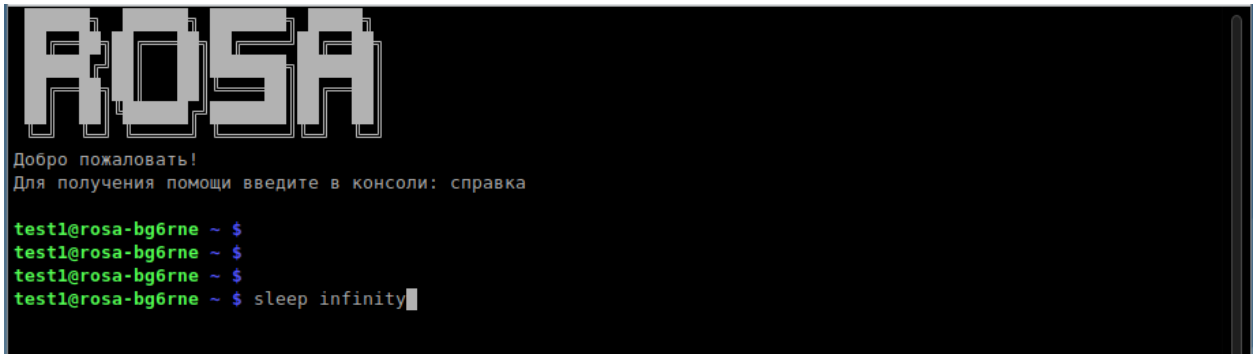


Рисунок 126 – Запуск процесса бесконечного ожидания

г) выйти из screen с помощью комбинации клавиш `Ctrl+A`, а затем `Ctrl+D`;

д) выйти из SSH-сессии одним из следующих способов:

– выполнить команду `exit`;
– использовать комбинацию клавиш `Ctrl+D`;
– закрыть консоль.

е) открыть новую SSH-сессию с помощью команды `ssh test1@192.168.122.76`;

ж) выполнить команду `screen -r`.

В результате выполнения команды `screen -r` видно, что начатый процесс бесконечного ожидания продолжает выполняться и после выхода пользователя из SSH-сессии (рисунок 127).

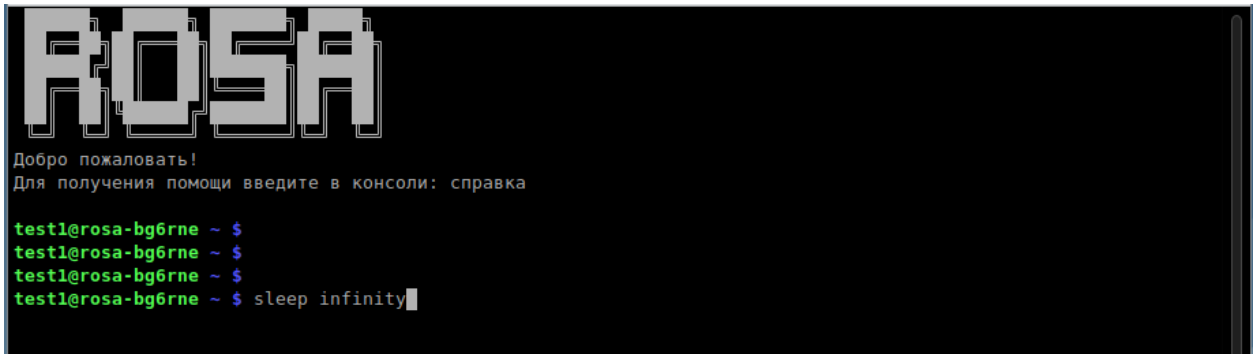


Рисунок 127 – Процесс бесконечного ожидания, сохраненный после выхода пользователя из SSH-сессии

Примечания – Пользователь может:

– Проверить наличие пакета в репозитории с помощью команды `dnf search systemd-settings` (рисунок 128);

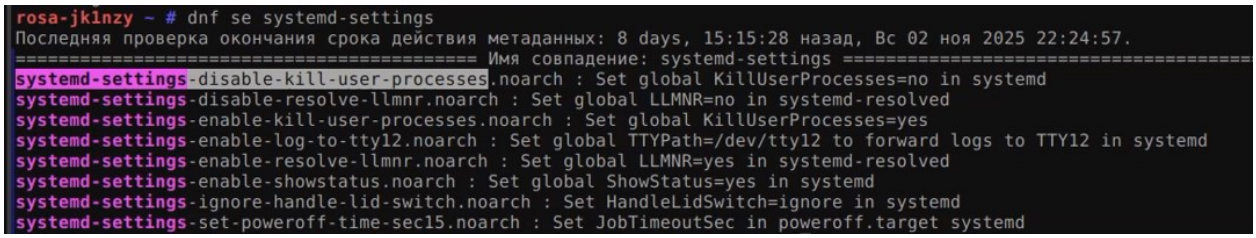


Рисунок 128 – Проверка наличия пакета systemd-settings-disable-kill-user-processes в репозитории

– Проверить содержимое пакета с помощью команды `rpm -ql systemd-settings-disable-kill-user-processes` (рисунок 129);



Рисунок 129 – Содержимое пакета systemd-settings-disable-kill-user-processes

– С правами суперпользователя выполнить команду `dnf remove systemd-settings-disable-kill-user-processes` для удаления пакета.

8.2 Терминальный доступ с использованием XRDP

Терминальный сервер XRDP предназначен для организации удалённого доступа пользователей к графическому сеансу рабочего стола через протокол

RDP. После подключения пользователь проходит аутентификацию и получает полноценную рабочую среду, аналогичную локальному сеансу.

Для установки терминального сервера XRDP необходимо выполнить установку следующих пакетов с помощью команды:

```
sudo dnf install xrdp task-x11
```

После установки требуется включить и запустить службу XRDP командой:

```
sudo systemctl enable --now xrdp
```

Для корректной работы терминального сеанса необходимо работать в одном из доступных графических окружений:

– Минимальное KDE Plasma 5; устанавливается командой:

```
sudo dnf install task-plasma5-minimal
```

– Полное KDE Plasma 5 с базовым набором программ; устанавливается командой:

```
sudo dnf install task-plasma5
```

– KDE Plasma 5, соответствующее десктопному образу ОС; устанавливается командой:

```
sudo dnf install task-iso-plasma5
```

– Минимальное XFCE; устанавливается командой:

```
sudo dnf install task-xfce-minimal
```

– XFCE с настройками РОСА и дополнительным ПО; устанавливается командой:

```
sudo dnf install task-xfce
```

– Окружение MATE; устанавливается командой:

```
sudo dnf install task-mate
```

– GNOME, соответствующее десктопному образу ОС; устанавливается командой:

```
sudo dnf install task-iso-gnome
```

8.2.1 Настройка локальных пользователей

Для предоставления доступа по RDP локальным пользователям необходимо:

а) создать локальную учётную запись:

```
sudo useradd <имя_пользователя>
```

б) задать пароль пользователю:

```
passwd <имя_пользователя>
```

в) добавить пользователя в группу tsusers, обладающую правом на запуск XRDP-сессий:

```
usermod -a -G tsusers <имя_пользователя>
```

8.2.2 Настройка доменных пользователей

Для обеспечения подключения доменных пользователей по протоколу RDP через XRDP необходимо настроить сопоставление доменных групп с локальной группой tsusers, используемой XRDP для контроля доступа:

а) Необходимо убедиться, что сервер введён в домен и Система успешно распознает доменные учетные записи. Проверить наличие информации о домене можно командой:

```
realm list
```

Если сервер введён в домен, в ответ будет показан список обнаруженных доменов.

Подробнее о вводе в домен см. раздел 6.3.5.

б) На контроллере домена добавить пользователей, которым требуется доступ по XRDP, в одну общую доменную группу, например, RDP-Users.

в) На сервере с ОС РОСА "ХРОМ" выполнить команду для сопоставления этой доменной группы с локальной группой tsusers:

```
sudo roleadd "Доменная_группа" tsusers
```

Например:

```
sudo roleadd "DOMAIN\\RDP-Users" tsusers
```

После выполнения этой команды все члены доменной группы RDP-Users будут автоматически считаться членами локальной группы tsusers и смогут подключаться к серверу через XRDP без необходимости вручную добавлять каждого пользователя в локальные группы.

г) Проверить корректность настроек командой:

```
id 'доменный_пользователь'
```

Вывод команды должен содержать группу tsusers.

8.2.3 Подключение с клиента

Для подключения к терминальному серверу можно использовать любой RDP-клиент, поддерживающий стандартный протокол. В качестве примера – клиент Remmina, доступный в официальном репозитории РОСА. Для установки Remmina в терминале нужно выполнить следующие действия:

а) обновить список пакетов:

```
sudo dnf makecache
```

б) установить пакет Remmina от имени пользователя root:

```
sudo dnf install remmina
```

в) после завершения установки запустить Remmina из меню приложений (категория Интернет) или выполнить в терминале:

```
remmina
```

Клиентское подключение в Remmina производится по IP-адресу терминального сервера с вводом имени пользователя и пароля, соответствующих разрешённой локальной или доменной учётной записи (рисунок 130).

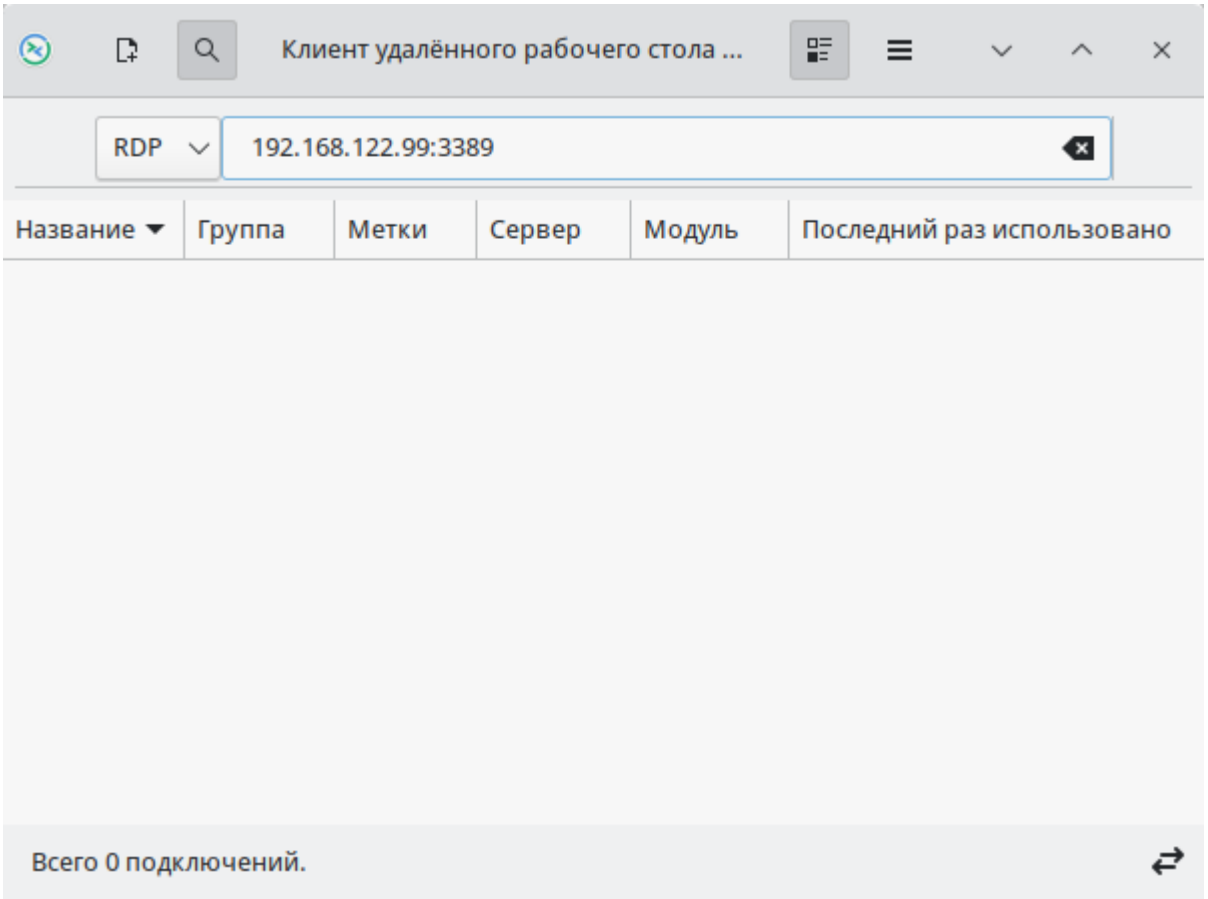


Рисунок 130 – Подключение по RDP с помощью клиента Remmina

9 АВТОМАТИЗАЦИЯ ИНФРАСТРУКТУРЫ

9.1 Написание bash-скриптов

9.1.1 Команды

Командный набор оболочки bash является одним из основных средств автоматизации административных задач и позволяет создавать сценарии для выполнения последовательности команд без участия пользователя. Опыт работы с командной строкой необходим для написания эффективных bash-сценариев. При увеличении объёма сценария свыше сотни строк или необходимости использования более сложных конструкций рекомендуется переходить на языки программирования Perl или Python.

Комментарии в сценариях начинаются со знака решетки (#) и продолжаются до конца строки. Одну логическую строку можно разделить на несколько физических, используя символ обратной косой черты (\). Несколько команд можно разместить в одной строке, разделив их точкой с запятой (;).

Пример простого сценария:

```
#!/bin/bash
echo "Hello, world!"
```

где #!/bin/bash указывает системе, что данный файл должен быть интерпретирован оболочкой bash, расположенной в каталоге /bin/.

Для выполнения сценария необходимо назначить файлу атрибут выполнения:

```
chmod +x helloworld
./helloworld
```

Результат:

```
Hello, world!
```

Если сценарий можно запустить без указания префикса ./, это означает, что текущий каталог (.) включён в переменную PATH. Такая настройка считается небезопасной, поскольку позволяет выполнить подменённые файлы при наличии прав записи в каталог.

Сценарий также можно выполнить, явно указав интерпретатор или используя встроенную команду:

```
bash helloworld
source helloworld
```

Первая команда запускает сценарий в новом экземпляре оболочки bash, а вторая выполняет его в текущей сессии.

Второй вариант используется в тех случаях, когда сценарий изменяет переменные окружения или выполняет операции, применимые только к текущей оболочке.

В Linux тип файла определяется не расширением, а установленными правами доступа. Для удобства допускается использовать суффикс .sh, однако при запуске команда должна включать это расширение, например:

```
bash script.sh
```

9.1.2 Сценарии

Перед переходом к особенностям написания bash-сценариев следует изучить методику этого процесса. Многие администраторы пишут bash-сценарии так же, как Perl- или Python-сценарии, т.е. используя какой-либо текстовый редактор, но рекомендуется для удобства рассматривать в качестве интерактивной среды разработки сценариев режим с приглашением на ввод команды.

В качестве примера можно рассмотреть наличие журналов регистрации, именованных с суффиксами .log и .LOG и размещенных по всей иерархической структуре каталогов, необходимость изменения этих суффиксы, приведя их наименования к прописному виду.

Сначала нужно отыскать все эти файлы:

```
$ find. -name '*log'

.do-not-touch/important.log
admin.com-log/
foo.log
genius/spew.log
leather_flog
```

Далее необходимо включить в шаблон точку и игнорировать при поиске каталоги, для чего нажать комбинацию клавиш `Ctrl+P`, чтобы вернуть команду в командную строку, а затем модифицировать ее.

```
$ find.-type f -name '*.log'

.do-not-touch/important.log
foo.log
genius/spew.log
```

С внесением таких изменений вывод команды становится более понятным. При необходимости можно избавиться от каталога .do-not-touch следующим образом:

```
$ find. -type f -name '*.log ' | grep -v.do-not-touch

foo.log
genius/spew.log
```

Теперь получен абсолютно точный список файлов, которые должны быть переименованы.

Далее можно попробовать сгенерировать несколько новых имен.

```
$ find. -type f -name '*.log ' I grep -v.do-not-touch | while
read fname
> do
> echo mv $ fname $ {fname/. log/. LOG/}
> done
mv foo.log foo.LOG
mv genius/spew.log genius/spew.LOG
```

Эти команды позволяют переименовать нужные файлы.

В реальности можно снова вызвать уже выполненную команду и отредактировать команду `echo`, чтобы заставить bash-оболочку выполнять команды `mv`, а не просто выводить их, так как передача команд в отдельную копию оболочки bash более надежный вариант работы, который к тому же требует меньшего объема редактирования.

При нажатии комбинации клавиш `Ctrl+P` bash-оболочка сворачивает мини-сценарий данного примера в одну. К этой уплотненной командной строке нужно добавить канал, передающий выходные данные команде `bash -x`.

```
$ find. -type f -name '*.log ' I grep -v.do-not-touch | while
read fname;
 do echo mv $fname $ {fname/. log/. LOG/}; done | bash -x

+ mv foo.log foo.LOG
+ mv genius/spew.log genius/spew.LOG
```

Ключ `-x` команды `bash` обеспечивает вывод каждой команды перед ее выполнением.

Теперь переименование файлов завершено, но полезно сохранить этот сценарий, чтобы воспользоваться им вновь. Встроенная в `bash` команда `fc` по

своему действию во многом аналогична нажатию комбинации клавиш `Ctrl+P`, но вместо возврата последней команды в командную строку она передает команду в заданный редактор. Следует добавить в нужный файл строку идентификационного комментария, поместить сохраненный файл в приемлемый каталог (например, ~/bin или /usr/local/bin), сделать файл исполняемым, чтобы в итоге получить настоящий сценарий.

Рекомендации по использованию сценариев:

– разработка сценария (или его части) должна быть в виде конвейера команд, причем пошагово и в режиме выполнения командных строк;

– пересылка результата должна осуществляться в стандартный выходной поток, проверяя правильность работы используемых команд;

– использовать на каждом этапе буфера ранее выполненных команд для их появления в командной строке и инструментов редактирования для их модификации;

– проводить сохранение всех команд сценария до тех пор, пока команды не заработают должным образом;

– проводить выполнение команды на реальном примере при получении правильного результата, чтоб убедиться в достоверности;

– использование команды `fc` для фиксации работы в редакторе, оформления ее соответствующим образом и сохранения.

В приведенном выше примере были выведены командные строки, а затем направлены в подоболочку для выполнения. Этот метод не является универсально применимым, но часто оказывается полезным. В качестве альтернативного варианта можно фиксировать результаты, перенаправляя их в файл.

Рекомендуется добиваться при разработке сценариев получения нужных результатов, и до этих пор не выполнять никаких потенциально деструктивных действий.

9.1.3 Ввод и вывод данных

Команда `echo` не отличается интеллектуальностью, но зато проста в применении. Для получения большего контроля над выводом данных используют команду `printf`, которая менее удобна, поскольку предполагает указание в нужных местах символов перехода на новую строку \n, но при этом позволяет использовать символ табуляции и другие средства форматирования результата.

Для сравнения результаты выполнения следующих двух команд:

```
$ echo "\taa\tbb\tcc\n"
\taa\tbb\tcc\n
```

```
$ printf "\taa\tbb\tcc\n"
aa bb cc
```

Для того чтобы сформировать пользователю приглашение ввести данные, можно использовать команду `read`:

```
#!/bin/bash
echo -n "Ввести свое имя: "
read user_name
if [ -n "$user_name" ]; then
 echo "Привет $user_name!"
 exit 0
else
 echo "Вы не назвали свое имя!"
 exit 1
fi
```

9.1.4 Функции и аргументы

Аргументы командной строки служат для сценария переменными с числовыми именами: $1 – первый аргумент командной строки, $2 – второй и т.д. Аргумент $0 содержит имя, по которому был вызван сценарий, например, ./bin/example.sh, т.е. это не фиксированное значение.

Переменная $# содержит количество переданных аргументов командной строки, а переменная $* все эти аргументы. Ни одна из этих переменных не учитывает аргумент $0.

При вызове сценария с некорректными аргументами или вообще без них должно быть выведено короткое сообщение с напоминанием о том, как следует использовать данный сценарий. Приведенный ниже сценарий, принимающий два аргумента, проверяет, являются ли оба эти аргумента каталогами, и, убедившись в этом, отображает их. Если аргументы оказываются недопустимыми, сценарий выводит сообщение о его правильном использовании и завершается с ненулевым значением кода возврата. Если программа, вызвавшая этот сценарий, проверит код возврата, она зафиксирует, что сценарий не удалось выполнить корректно.

```
#!/bin/bash

function show_usage {
 echo "Использование: $0 source_dir dest_dir"
 exit 1
}
```

```
# Основная программа начинается здесь

if [ $# -ne 2 ]; then
 show_usage
else # Существуют два аргумента
 if [ -d $1 ]; then
 source_dir=$1
 else
 echo 'Недопустимый каталог-источник'
 show_usage
 fi
 if [ -d $2 ]; then
 dest_dir=$2
 else
 echo 'Недопустимый каталог-приемник'
 show_usage
 fi
fi

printf "Каталог-источник: ${source_dir}\n"
printf "Каталог-приемник: ${dest_dir}\n"
```

Для вывода сообщения о правильном использовании данного сценария была создана отдельная функция show_usage. Если бы позже этот сценарий был модифицирован и стал бы принимать дополнительные аргументы, это сообщение нужно было бы изменить только в одном месте.

```
$ mkdir aaa bbb
$ sh showusage aaa bbb
Каталог-источник: aaa
Каталог-приемник: bbb
$ sh showusage foo bar
Недопустимый каталог-источник
Использование: showusage source__dir dest_dir
```

Аргументы bash-функций обрабатываются практически так же, как и аргументы командной строки: $1 – первый аргумент командной строки, $2 – второй и т.д. Как видно из приведенного выше примера, аргумент $0 содержит имя сценария.

Для того чтобы сделать предыдущий пример более устойчивым к ошибкам, можно передавать функции show_usage в качестве аргумента код ошибки. Это

позволило бы конкретизировать код возврата для каждого типа отказа. Реализация этой идеи показана в следующем фрагменте программы:

```
function show_usage {
 echo "Использование: $0 source_dir dest_dir"
 if [ $# -eq 0 ]; then
 exit 99 # Выход с любым ненулевым кодом возврата
 else
 exit $1
 fi
}
```

В этой версии функции добавляется анализ наличия аргументов. Внутри функции переменная $# сообщает, сколько аргументов было ей передано. Сценарий завершится с кодом возврата 99, если при его вызове не было передано никаких аргументов. Но при передаче конкретного значения, например, "show_usage 5", сценарий завершится с этим кодом после вывода сообщения об использовании сценария (переменная $? содержит статус завершения последней выполненной команды, независимо от факта использования в сценарии или в командной строке).

Между функциями и командами в bash-оболочке существует строгая аналогия. Можно определить полезные функции в файле ~/.bash_profile, а затем использовать их в командной строке как команды. Например, если узел стандартизировал в сети порт 7988 для протокола SSH, можно определить в файле ~/.bash_profile функцию ssh, чтобы она всегда запускалась (как команда) с ключом -p 7988.

```
function ssh {
 /usr/bin/ssh -p 7988 $*
}
```

Подобно многим командным оболочкам, bash обладает механизмом псевдонимов (alias), который может репродуцировать этот ограниченный пример в еще более лаконичном виде. Тем не менее, функции остаются более универсальным и более мощным средством.

9.1.5 Области видимости и потоки

Переменные в рамках сценария имеют глобальный статус, но функции могут создавать собственные локальные переменные с помощью объявления local.

Следующий код используется для примера:

```
    #!/bin/bash

    function localizer {
     echo "==> В функции localizer начальное значение a равно
'$a' "
     local a
     echo "==> После объявления local значение a стало равным
'$a' "
     a="localizer version"
     echo "==> При выходе из функции localizer значение a равно
'$a' "
    }
    a="test"
    echo "До вызова функции localizer значение a равно '$a' "
    localizer
    echo "После вызова функции localizer значение a равно '$a' "
```

По приведенным ниже результатам выполнения сценария scopetest.sh видно, что локальная версия переменной $a внутри функции localizer перезаписывает глобальную переменную $a. Глобальная переменная $a видна в функции localizer до тех пор, пока не встретится объявление local, т.е. объявление local работает как команда, которая в момент выполнения создает локальную переменную.

При запуске скрипта:

```
    $ sh scopetest.sh
```

До вызова функции localizer значение a равно "test":

```
    ==> В функции localizer начальное значение a равно "test"
    ==> После объявления local значение a стало равным ""
    ==> При выходе из функции localizer значение a равно "localizer
version".
    После вызова функции localizer значение a равно "test".
```

9.1.6 Операторы

Терминатором (признаком конца) для условного оператора if служит оператор fi. Для образования цепочки if-операторов можно использовать ключевое слово elif, означающее "else if".

```
    if [ $base -eq 1 ] && [ $dm -eq 1 ]; then
     installDMBase
```

```
elif [ $base -ne 1 ] && [ $dm -eq 1 ]; then
 installBase
elif [ $base -eq 1 ] && [ $dm -ne 1 ]; then
 installDM
else
 echo '==> Installing nothing'
fi
```

Как и специальный [] -синтаксис для выполнения операций сравнения, так и ключеподобные имена операторов целочисленного сравнения (например, -eq) происходят от использования утилиты /bin/test старых оболочек. В действительности квадратные скобки – это не что иное, как условное обозначение вызова утилиты test; они не являются частью оператора if.

Ниже в таблице 12 собраны bash-операторы сравнения для чисел и строк. В отличие от Perl, в bash используются текстовые операторы для чисел и символические операторы для строк.

Таблица 12 - Операторы сравнения чисел и строк

Строки	Числа	Истина, если
x = y	x -eq y	x равно y
x != y	x -ne y	x не равно y
x < y	x -lt y	x меньше y
x <= y	x – le y	x меньше или равно y
x > y	x – gt y	x больше y
x >=y	x -ge y	x больше или равно y
-n x	-	x не равно значению null
-n x	-	x равно значению null

В bash-оболочке предусмотрены возможности оценки свойств файлов. Некоторые из операторов тестирования и сравнения файлов приведены ниже в таблице 13 .

Таблица 13 - Операторы оценки файлов

Оператор	Истина, если
-d файл	Файл файл существует и является каталогом
-e файл	Файл файл существует
-f файл	Файл файл существует и является обычным

Оператор	Истина, если
-r файл	У вас есть право доступа для чтения файла
-s файл	Файл файл существует, и он не пустой
-w файл	У вас есть право доступа для записи в файл
файл1 -nt файл2	Файл файл1 новее, чем файл2
файл1 -ot файл2	Файл файл1 старше, чем файл2

Несмотря на всю полезность формы elif, лучше (с точки зрения ясности программного кода) использовать case-структуру выбора варианта. Ниже показан ее синтаксис на примере функции, которая централизирует процесс регистрации сообщений для сценария. Конкретные варианты описываются закрывающими скобками после каждого условия и двумя точками с запятой, завершающими блок операторов, который должен быть выполнен при реализации заданного условия. Оператор case завершается ключевым словом esac.

```
    # Уровень протоколирования устанавливается в глобальной
    # переменной LOG_LEVEL. Возможные варианты перечислены в
порядке
    # от самого строгого до наименее строгого: Error, Warning,
Info и Debug.
    function logMsg {
     message_level-$1
     message_itself=$2
     if [ $message_level -le $LOG_LEVEL ]; then
     case $message_level in
     0) message_level_text="Error" ;;
     1) message_level_text="Warning" ;;
     2) message_level_text="Info" ;;
     3) message_level_text="Debug" ;;
     *) message_level_text="Other"
     esac
     echo "${message_level_text}: $message_itself"
     fi
    }
```

Эта функция иллюстрирует общепринятую парадигму уровня регистрации (log level), используемую многими приложениями административного характера. Код этого сценария позволяет генерировать сообщения на различных уровнях

детализации, но действительно регистрируются или отрабатываются только те из них, которые проходят глобально устанавливаемый порог $LOG_LEVEL. Чтобы прояснить важность каждого сообщения, его текст предваряется меткой, описывающей соответствующий уровень регистрации.

9.1.7 Циклы и конструкции

Конструкция for...in предназначена для упрощения выполнения некоторых действий для группы значений или файлов, особенно при универсализации файловых имен, т.е. замене реальных символов в имени и расширении универсальными (например, "*" и "?") с целью формирования целых списков имен файлов. Шаблон *.sh в приведенном ниже цикле for позволяет обработать целый список совпадающих с ним (шаблоном) имен файлов из текущего каталога. Оператор for, проходя по этому списку, по очереди присваивает имя каждого файла переменной $file.

```
#!/bin/bash
suffix=BACKUP--`date +%Y%m%d-%H%M`
for script in *.sh; do
 newname="$script.$suffix"
 echo "Copying $script to $newname..."
 cp $script $newname
done
```

Результат выполнения этого сценария таков:

```
$ sh forexample
Copying rhel.sh to rhel.sh.BACKUP--20091210-1708...
Copying sles.sh to sles.sh.BACKUP--20091210-1708...
```

В раскрытии имени файла все работает в точном соответствии с тем, как написано в командной строке. Другими словами, сначала имя файла раскрывается (т.е. шаблон заменяется существующим именем), а затем уже обрабатывается интерпретатором в развернутом виде.

Аналогично можно ввести имена файла статически, как показано в командной строке:

```
for script in rosa.sh postinst.sh; do
```

В действительности любой список имен, содержащих пробельные символы (включая содержимое переменной), обрабатывается как объект циклической конструкции for...in.

Bash-оболочке также присуща близость к циклу for из традиционных языков программирования, в которых задается стартовое выражение, инкрементация и условие окончания цикла.

```
for (( i=0 ; i < $CPU_COUNT ; i++ )) ; do
CPU_LIST="$CPU_LIST $i"
done
```

На примере следующего сценария иллюстрируется bash-цикл while, который часто применяется для обработки аргументов командной строки и чтения строк файла:

```
#!/bin/bash
exec 0<$1
counter=l
while read line; do
echo "$counter: $line"
$((counter++))
done
```

Результат выполнения этого сценария:

```
rosa$ sh whileexainple /etc/passwd
1: root:x:0:0:Superuser:/root:/bin/bash
2: bin:x:l:l:bin:/bin:/bin/bash
3: daemon:x:2:2:Daemon:/sbin:/bin/bash
```

В этом сценарии есть интересные особенности. Оператор exec переопределяет стандартный выходной поток сценария так, чтобы входные данные считывались из файлов, имена которых должны определяться в соответствии с первым аргументом командной строки. Если окажется, что такой файл не существует, данный сценарий сгенерирует ошибку.

Оператор read в конструкции while на самом деле является встроенным в оболочку, но действует подобно внешней команде. Внешние команды можно также помещать в конструкцию while. Цикл while в такой форме завершится при условии, если внешняя команда возвратит ненулевое значение кода завершения.

Конструкция $((counter++)) обозначает вычисление выражения. Кроме того, оно делает необязательным использование символа "$" для обозначения имен переменных. Удвоенный знак "++" – это оператор постинкремента, знакомый, например, по языку C. Он возвращает значение переменной, с которой он связан, но также увеличивает на 1 это значение.

Выражения $((...)) работают в контексте двойных кавычек, поэтому тело рассмотренного выше цикла можно свернуть до одной строки.

```
while read line; do
echo "$((counter++)) : $line"
done
```

9.1.8 Арифметические операции и массивы

Сложные структуры данных и вычисления нельзя отнести к сильной стороне bash-оболочки, но применяются работа с массивами и арифметическими операциями.

Все bash-переменные представляют собой строковые значения, поэтому bash-оболочка не делает различия в присваиваниях между числом 1 и символьной строкой 1. Различие лежит в использовании переменных. Следующий код иллюстрирует это различие:

```
#!/bin/bash

a=1
b=$((2))

c=$a+$b
d=$(($a+$b))

echo "$a + $b = $c \t(знак плюс как строковый литерал)"
echo "$a + $b = $d \t(знак плюс как арифметическое сложение)"
```

При выполнении этого сценария получится такой результат:

```
1 + 2 = 1+2 (знак плюс как строковый литерал)
1 + 2 = 3 (знак плюс как арифметическое сложение)
```

Следует обратить внимание, что знак "плюс" в присваивании переменной $c не работает как оператор конкатенации для строк. Он остается всего лишь литеральным символом, поэтому эта строка эквивалентна следующей:

```
c="$a+$b"
```

Для того чтобы добиться вычисления, необходимо заключить выражение в двойные скобки $((...)), как показано выше в присваивании переменной $d. Но даже эта мера предосторожности не позволяет получить в переменной $d числового значения; это значение по-прежнему хранится в виде строки 3.

В оболочке bash реализован обычный ассортимент операторов: арифметических, логических и отношений (подробнее см. соответствующие man-страницы).

Массивы в командной bash-оболочке могут показаться немного странными объектами и используются не очень часто. Однако при необходимости их можно применять. Литеральные массивы ограничиваются круглыми скобками, а отдельные элементы разделяются пробельными символами. Для включения литеральных пробелов в элемент можно использовать кавычки:

```
example=(aa 'bb cc' dd)
```

Для доступа к отдельным элементам массива используют выражение ${имя_ массива [индекс]}. Индексация начинается с нуля. Такие индексы, как "*" и "@", относятся к массиву в целом, а специальные конструкции ${#имя_массива[*] } и ${#имя_массива[@] } возвращают количество элементов в массиве. Не следует путать эти выражения с конструкцией ${#имя_массива} хотя эта форма кажется более логичной, но в действительности она содержит указатель на первый элемент массива (эквивалент для ${#имя_массива[0]}).

Можно подумать, что выражение $example[1] должно служить однозначной ссылкой на второй элемент массива, но bash-оболочка интерпретирует эту строку так: $example (обозначение ссылки на $example[0]) плюс литеральная строка [1]. Отсюда вывод: ссылаясь на переменные массива, всегда используют фигурные скобки (без каких-либо исключений).

В качестве примера можно рассмотреть сценарий, который иллюстрирует некоторые особенности bash-массивов и подводные камни, на которые можно наткнуться при управлении ими.

```
#!/bin/bash

 example=(aa 'bb cc' dd)
 example[3]=ee

 echo "example[@] = ${example[0]}"
 echo "Массив example содержит ${#example[@]} элементов"

for elt in "${example[8] }"; do
 echo " Элемент = $elt"
 done
```

Так выглядит результат выполнения этого сценария.

```
$ sh arrays
example [@] = aa bb cc dd ee
Массив example содержит 4 элемента
 Элемент = aa
```

```
 Элемент = bb cc
 Элемент = dd
 Элемент = ee
```

Данный пример правильно реализован и поэтому выглядит не сложным к выполнению. Однако, сценарий с заменой for-строки вариантом (т.е. без кавычек, в которых заключено выражение массива) также работает, но вместо четырех элементов он выведет пять: aa, bb, cc, dd и ee:

```
for elt in ${example[@]}; do
```

Важно – все bash-переменные все равно остаются строками, поэтому работа с массивами в некотором роде иллюзия.

9.2 Ansible

Ansible – это инструмент для автоматизации управления конфигурациями, развёртывания программ и выполнения сложных задач. Он помогает Администраторам управлять большим количеством серверов и хостов централизованно, обеспечивая простоту и эффективность процессов. Ansible интегрируется в архитектуру ОС и поддерживает автоматизацию множества операций.

Ansible использует архитектуру без агента, работая по SSH-протоколу, что упрощает настройку и использование. Все операции описываются в виде "плейбуков" на языке YAML, что делает управление конфигурациями прозрачным и легко читаемым.

Основные компоненты Ansible:

– Модули – это основные рабочие единицы Ansible, которые выполняют определённые задачи, такие как управление пакетами, работой с файлами и сетевыми интерфейсами.

– Инвентарь (Inventory) – список узлов (хостов), к которым Ansible будет обращаться для выполнения задач. Инвентарь может быть статическим (описан в текстовом файле) или динамическим (автоматически генерируется).

– Плейбуки (Playbooks) – наборы инструкций, написанных на YAML, определяющие, какие задачи нужно выполнить на целевых хостах.

– Роли (Roles) – механизм, который группирует плейбуки, переменные и задачи для более эффективного и модульного управления инфраструктурой.

9.2.1 Установка и подключение хостов

Для установки Ansible нужно выполнить команду от имени root:

```
sudo dnf install ansible
```

Для подключения хостов к Системе автоматизации следует выполнить следующие шаги:

а) В файле /etc/ansible/hosts прописать все хосты, на которые будет распространяться конфигурация. Хосты можно разделить по группам, а также, если есть домен, автоматически экспортировать список из домена. Можно прописывать как IP-адреса, так и имена хостов, если они видны DNS-сервером в сети. Для тестового примера можно прописать два хоста:

```
[test]
10.10.1.111
10.10.1.74
```

б) Подключение к хостам осуществляется по SSH-протоколу с помощью RSA-ключей; сгенерировать серверный ключ можно командой (в процессе ее выполнения следует на все запросы нажимать клавишу `Enter`):

```
ssh-keygen -C "$(whoami)@$(hostname)-$(date -I)"
```

в) Распространить ключ на все подключенные хосты командой:

```
ssh-copy-id root@server
```

где:

– root – пользователь, от имени которого будут выполняться плейбуки;
– server – IP-адрес хоста.

Пример: root@10.10.1.74.

9.2.2 Проверка подключения

Для проверки подключения удаленных хостов с помощью Ansible нужно выполнить команду:

```
ansible test -m ping
```

где test – это группа хостов, указанная в файле hosts.

В выводе команды для каждого хоста должен присутствовать успешный результат:

```
 test-1 | success >> {
 "changed": false, U
 "ping": "pong"
 }

 test-2 | success >> {
 "changed": false,
```

```
 "ping": "pong"
 }
```

9.2.3 Описание сценариев Ansible

Для работы с Ansible на клиенте достаточно SSH-соединения. Задача Ansible – выполнить одновременно ряд операций на нескольких удаленных рабочих станциях. Все операции, которые выполняются последовательно, записываются в некий Ansible-скрипт – плейбук, основанный на собственном наборе команд. Записи ведутся в простом формате и построены на YAML-разметке. Пример записи сценария:

```
- name: "First step"
 hosts: localhost
 tasks:
 - taskA
 - taskB
```

где:

– name: "First step" – имя группы задач;
– hosts: localhost – хосты, на которых будут выполнены задачи;
– параметры, перечисленные после команды `tasks` – перечень задач, которые необходимо выполнить.

В одном плейбуке может быть несколько блоков с задачами, для чего необходимо добавить дополнительные параметры name после завершения описания предыдущей группы задач.

Команды программы – tasks – являются модулями Ansible. Таким образом, задавая параметры task, вызывается модуль Ansible с его параметрами. Записать task можно в двух видах: кратком и полном.

Пример команды установки curl при помощи dnf-модуля в кратком виде:

```
-dnf: name=curl state=latest
```

Тогда плейбук с этим task будет выглядеть следующим образом:

```
- name: "First step"
 hosts: localhost
 tasks:
- dnf: name=curl state=latest
```

Возможность более полно описать команду `task` – присвоить ей имя (name). Тогда получившийся текст выглядит как подзадача основного блока задач:

```
- name: "First step"
 hosts: localhost
 tasks:
 - name: "Add curl package"
 dnf: name=curl state=latest
```

В этом случае сразу видно, что есть блок задач "First step", в нем существует одна задача – установка пакета curl.

9.2.4 Создание сценариев

Плейбуки являются сценариями, выполняемыми на удаленных хостах. В качестве примера можно создать плейбук, устанавливающий программы, выполнением следующих действий:

а) Создать каталог для хранения плейбуков, которые пишутся на языке YAML:

```
sudo mkdir /etc/ansible/playbooks
```

б) В каталоге /etc/ansible/playbooks создать файл install_programm.yml:

```
sudo touch /etc/ansible/playbooks/install_programm.yml
```

Файл имеет следующее содержимое:

```
---
- hosts: all
tasks:
- name: Install programm
dnf:
name: "{{ packages }}"
vars:
packages:
- vim
- zstd
```

В данном примере для установки на всех хостах используется модуль dnf и массив packages, который содержит подлежащие установке программы.

Для определения хостов, на которых будет проведена установка, можно воспользоваться командой:

```
ansible-playbook /etc/ansible/playbooks/install_programm.yml
--list-host
```

Созданный набор инструкций можно запустить командой:

```
ansible-playbook /etc/ansible/playbooks/install_programm.yml
```

9.2.5 Выполнение одиночных команд

С помощью Ansible возможно сразу на всех клиентах выполнить команду `bash` без создания плейбуков. Например, чтобы просмотреть информацию об использовании оперативной памяти на удаленных хостах, нужно выполнить команду:

```
ansible test -a "free -h"
```

10 КОНТЕЙНЕРИЗАЦИЯ

Контейнеризация – это метод виртуализации на уровне ОС, который позволяет запускать приложения и их зависимости в изолированных контейнерах. Контейнеры включают в себя все необходимые для работы приложения компоненты, такие как библиотеки, зависимости и настройки, но используют ядро ОС хоста, что делает их легковесными по сравнению с виртуальными машинами.

Контейнеризация является отличной альтернативой аппаратной виртуализации. Преимущества использования контейнеров:

– Изоляция окружения: контейнеры гарантируют, что приложения не будут конфликтовать друг с другом по зависимостям и настройкам;

– Упрощение развертывания: контейнеры можно развертывать одинаково в разных средах, что снижает риск ошибок, связанных с различиями в окружениях;

– Использование контейнеров позволяет легко наращивать или сокращать количество работающих экземпляров приложения;

– Контейнеры используют общие ресурсы ОС, что снижает нагрузку на оборудование.

В Системе используются такие популярные технологии контейнеризации, как Docker и Kubernetes. Далее можно рассмотреть основные принципы работы с данным ПО.

10.1 Docker

Одним из наиболее популярных инструментов для программной виртуализации является программа Docker – автоматизированное средство управления виртуальными контейнерами. Он решает множество задач, связанных с созданием контейнеров, размещением в них приложений, управлением процессами, а также тестированием ПО и его отдельных компонентов.

Docker – программное обеспечение с открытым исходным кодом, применяемое для разработки, тестирования, доставки и запуска веб-приложений в средах с поддержкой контейнеризации. Он нужен для более эффективного использования Системы и ресурсов, быстрого развертывания готовых программных продуктов, а также для их масштабирования и переноса в другие среды с гарантированным сохранением стабильной работы.

Docker-контейнеры работают в разных средах: локальном центре обработки информации, облаке, персональных компьютерах и т. д.

Docker представляет собой набор нескольких основных компонентов:

– Docker-демон (Docker-daemon) – сервер контейнеров, входящий в состав программных средств Docker. Демон управляет Docker-объектами (сетями, хранилищами, образами и контейнерами). Демон также может связываться с другими демонами для управления сервисами Docker.

– Docker-клиент (Docker-client / CLI) – интерфейс взаимодействия пользователя с Docker-демоном. Клиент и Демон – важнейшие компоненты "движка" Докера (Docker Engine). Клиент Docker может взаимодействовать с несколькими демонами.

– Docker-образ (Docker-image) – файл, включающий зависимости, сведения, конфигурацию для дальнейшего развертывания и инициализации контейнера.

– Docker-файл (Docker-file) – описание правил по сборке образа, в котором первая строка указывает на базовый образ. Последующие команды выполняют копирование файлов и установку программ для создания определенной среды для разработки.

– Docker-контейнер (Docker-container) – это легкий, автономный исполняемый пакет программного обеспечения, который включает все необходимое для запуска приложения: код, среду выполнения, системные инструменты, системные библиотеки и настройки.

– Том (Volume) – эмуляция ФС для осуществления операций чтения и записи. Она создается автоматически с контейнером, поскольку некоторые приложения осуществляют сохранение данных.

– Реестр (Docker-registry) – зарезервированный сервер, используемый для хранения docker-образов. Примеры реестров:

– Центр Docker – реестр, используемый для загрузки docker-image. Он обеспечивает их размещение и интеграцию с GitHub и Bitbucket.

– Контейнеры Azure – предназначен для работы с образами и их компонентами в директории Azure (Azure Active Directory).

– Доверенный реестр Docker или DTR – служба docker-реестра для инсталляции на локальном компьютере или сети компании.

– Docker-хаб (Docker-hub) или хранилище данных – репозиторий, предназначенный для хранения образов с различным ПО. Наличие готовых элементов влияет на скорость разработки.

– Docker-хост (Docker-host) – машинная среда для запуска контейнеров с программным обеспечением.

– Docker-сети (Docker-networks) – применяются для организации сетевого интерфейса между приложениями, развернутыми в контейнерах.

– Docker Engine ("движок" Docker) – ядро механизма Docker. "Движок" отвечает за функционирование и обеспечение связи между основными Docker-объектами (реестром, образами и контейнерами).

Сервер выполняет инициализацию демона (фоновой программы), который применяется для управления и модификации контейнеров, образов и томов.

REST API – механизм, отвечающий за организацию взаимодействия Docker-клиента и Docker -демона.

Клиент – позволяет пользователю взаимодействовать с сервером при помощи команд, набираемых в интерфейсе (CLI).

Установка и запуск сервиса docker:

```
sudo dnf install docker docker-containerd docker-compose -y
sudo systemctl start docker
sudo systemctl enable docker
```

10.2 Podman

10.2.1 Общие сведения

Podman — это инструмент контейнеризации, соответствующий спецификациям OCI (Open Container Initiative), предназначенный для создания, запуска и управления контейнерами без использования фонового демона. В отличие от других контейнерных платформ, использующих центральный управляющий процесс (демон), Podman работает по принципу daemonless и запускает контейнеры как обычные процессы ОС. Такой подход повышает безопасность среды и упрощает контроль над контейнерными задачами.

Podman также обеспечивает полную совместимость с Docker, на уровне CLI, форматов образов и Dockerfile что позволяет использовать большинство команд Docker при повышенном уровне изоляции и управляемости.

Podman может работать как от имени пользователя root, так и от имени обычного пользователя (rootless-режим).

Каждый контейнер запускается как отдельный процесс без зависимости от единого управляющего демона. Это обеспечивает большую отказоустойчивость: завершение Podman не приводит к остановке уже запущенных контейнеров.

Особенности использования Podman:

– rootful-режим для системных контейнеров;
– использование внешних registry (Docker Hub, Quay, GitHub Container Registry);
– применение расширенных сетевых моделей;
– интеграцию с Kubernetes через генерацию pod manifests;
– работу через Podman API Socket и контейнерный REST API.

Podman является рекомендуемым инструментом контейнеризации в следующих случаях:

– при необходимости строгого соблюдения политик безопасности;
– на серверах, где нежелательно наличие фоновых демонов;
– в системах с повышенными требованиями к контролю доступа;
– при работе пользователей без привилегий root;
– для изолированного запуска сервисов или тестирования ПО;
– при подготовке контейнеров для Kubernetes / OpenShift (поддержка pack/push в OCI registry).

Основные компоненты архитектуры Podman:

– runc/crun — низкоуровневые исполнители контейнеров, соответствующие стандарту OCI Runtime;
– conmon — вспомогательный процесс, отслеживающий состояние контейнера, фиксацию вывода и корректное завершение;
– containers/storage и containers/image — подсистемы хранения, обеспечивающие работу с образами и файловыми системами контейнеров;
– Buildah — компонент сборки образов, используемый Podman для выполнения операций build без демона;
– Skopeo — инструмент для копирования и проверки образов в локальных и удалённых реестрах.

Podman использует механизмы безопасности ядра Linux и обеспечивает запуск контейнеров в двух режимах:

– Rootless-режим:

  – контейнеры выполняются от имени обычного пользователя;
  – пользовательские пространства имён ограничивают видимость uid/gid;
  – запрещён доступ к привилегированным возможностям ядра;
  – повышенная изоляция и минимизация риска компрометации Системы.

Рекомендуется придерживаться принципов минимизации прав контейнеров и применения rootless-режима, если это возможно;

– Rootful-режим:

  – контейнеры запускаются от имени суперпользователя;
  – поддерживаются расширенные сетевые модели, дополнительные возможности ядра и повышенные привилегии;
  – используется только для специализированных задач администрирования.

Основные механизмы изоляции, применяемые Podman, соответствует принципам минимального привилегирования:

– User namespaces — изоляция пользовательских и групповых идентификаторов;

– Mount namespaces — разделение файловых систем;

– PID, IPC и Network namespaces — изоляция процессов, межпроцессного взаимодействия и сетевой среды;

– cgroups v2 — контроль потребления ресурсов;

– seccomp — фильтрация системных вызовов;

– nftables — основной механизм контроля сетевых политик на уровне хоста.

10.2.2 Установка и базовая настройка

Podman входит в стандартные репозитории ОС и устанавливается штатными средствами пакетного менеджера. Для установки Podman и сопутствующих инструментов необходимо выполнить команду:

```
sudo dnf install podman buildah skopeo
```

10.2.2.1 Настройка rootless-режима (рекомендуемый режим)

Режим работы от обычного пользователя (rootless-режим) является рекомендуемым режимом для работы. Для его настройки необходимо выполнить следующие шаги:

а) Подготовка пользователя. Необходимо убедиться, что для пользователя доступны UID/GID-диапазоны. Для проверки выполняются команды:

```
cat /etc/subuid
cat /etc/subgid
```

Если записи отсутствуют, их следует добавить вручную:

```
sudo usermod --add-subuids 100000-165535 <имя_пользователя>
sudo usermod --add-subgids 100000-165535 <имя_пользователя>
```

б) Проверка запуска контейнера от пользователя:

```
podman run --rm alpine echo "Rootless OK"
```

Особенности rootless-режима:

– является основным и рекомендуемым способом эксплуатации контейнеров;

– использование сети осуществляется через slirp4netns, что исключает необходимость привилегий;

– контейнеры изолированы пространствами имён и не могут воздействовать на системные ресурсы вне своего диапазона.

10.2.2.2 Настройка rootful-режима

Rootful-режим допускается для выполнения задач, требующих доступа к системной сети, использованию привилегированных контейнеров или интеграции с инфраструктурными сервисами.

Запуск контейнера:

```
sudo podman run --rm alpine id
```

где:

– alpine – имя образа из контейнерного реестра, на основе которого будет создан контейнер;

– id – команда, которая выполняется внутри контейнера;

– --rm – опция удаления контейнера после завершения

Таким образом, Alpine в данном случае является базовым образом, внутри которого выполняется команда `id`, чтобы продемонстрировать работу rootful-режима.

10.2.2.3 Настройка хранилища контейнеров

Podman использует подсистему containers/storage. Конфигурация располагается в:

– для rootless-режима в ~/.config/containers/storage.conf

– для rootful-режима в /etc/containers/storage.conf

Основной параметр:

```
graphroot="/var/lib/containers/storage"
```

При необходимости хранилище может быть вынесено в отдельный раздел.

Базовая сетевая конфигурация:

– в rootless-режиме – используется пользовательский сетевой стек slirp4netns. Проверка доступности сети из контейнера:

```
podman run --rm alpine ping -c2 8.8.8.8
```

– в rootful-режиме – создаются виртуальные мостовые интерфейсы (cni-podmanX). Файл конфигурации /etc/cni/net.d/87-podman-bridge.conflist.

Управление сетевыми политиками контейнеров следует выполнять через системные правила на хосте (nftables), а не внутри контейнеров.

10.2.3 Работа с контейнерами Podman

10.2.3.1 Запуск контейнеров

Простой запуск контейнера осуществляется по команде:

```
podman run --rm alpine echo "Hello from Podman"
```

где:

– run — запуск контейнера;
– --rm — удаление контейнера после завершения;
– alpine — имя образа.

Если образ ранее не был загружен в локальное хранилище, Podman автоматически скачает его из контейнерного реестра.

Проверка установленного окружения осуществляется командой:

```
podman run --rm registry.access.redhat.com/ubi9/ubi cat /etc/os-release
```

10.2.3.2 Управление образами

Основные параметры по управлению образами:

– загрузка образа (pull):

```
podman pull registry.access.redhat.com/ubi9/ubi
```

– просмотр локальных образов:

```
podman images
```

– удаление образа:

```
podman rmi <ID_или_имя_образа>
```

– проверка подписи образа:

```
skopeo inspect --raw docker://<реестр/образ> | jq .
```

10.2.3.3 Управление контейнерами

Основные параметры по управлению контейнерами:

– просмотр всех контейнеров:

```
podman ps -a
```

– запуск и остановка:

```
podman start <ID>
podman stop <ID>
```

– удаление контейнера:

```
podman rm <ID>
```

– получение журнала контейнера

```
podman logs <ID>
```

– выполнение команды в работающем контейнере

```
podman exec -it <ID> bash
```

10.2.3.4 Управление томами и данными

Основные параметры по управлению томами и данными:

– создание тома:

```
podman volume create data1
```

– запуск контейнера с подключенным томом:

```
podman run -v data1:/var/data alpine ls /var/data
```

– просмотр списка томов:

```
podman volume ls
```

Не рекомендуется размещать тома в разделах, исключённых из проверок безопасности (например, tmpfs без атрибутов контроля).

10.2.3.5 Работа с сетями контейнеров

Сетевое взаимодействие контейнеров Podman зависит от режима запуска — rootless или rootful:

– Rootless-режим (по умолчанию). В данном режиме контейнеры запускаются без привилегий и не имеют прямого доступа к сетевому стеку хоста:

– для организации сети используется slirp4netns, обеспечивающий изоляцию и автоматический NAT;

– публикация портов выполняется командой:

```
podman run --rm -p 8080:80 nginx
```

Rootless-сеть не изменяет системные сетевые настройки и считается безопасной для многопользовательских сред.

– Rootful-режим. При использовании rootful-контейнеров Podman применяет сетевую подсистему CNI; конфигурация сетей хранится в каталоге /etc/cni/net.d/.

– просмотр доступных сетей:

```
podman network ls
```

– создание пользовательской сети:

```
podman network create mynet
```

CNI-сети позволяют изолировать контейнеры и использовать отдельные подсети.

10.2.3.6 Ресурсные ограничения

Podman поддерживает управление ресурсами CPU и памяти через cgroups.

Формат задания ограничений:

```
podman run <ограничения> <образ>
```

– **Ограничения памяти (Memory limits):**

– ограничение объёма оперативной памяти: --memory=512m;

– жёсткое ограничение, запрещающее swap: --memory-swap=512m; если значение совпадает с параметром --memory, тогда swap запрещён.

– разрешённый объём swap: --memory-swap=1g;

– минимально гарантируемый объём памяти (soft limit): --memory-reservation=256m; контейнер может превысить ограничение, если у хоста достаточно ресурсов.

– ограничение скорости операций чтения/записи (throttling). Чтение: --memory-swappiness=0, где 0 — запрет использования swap.

– **Ограничения CPU:**

– вычислительная мощность, количество CPU для контейнера: --cpus=1.5;

– ограничение доли CPU (CPU quota): --cpu-quota=50000;

– CPU period (обычно менять не требуется): --cpu-period=100000;

– ограничение CPU в процентах: --cpu-percent=30;

– привязка контейнера к определённым ядрам CPU: --cpuset-cpus=0,2,4;

– привязка к группам процессорных сокетов/NUMA: --cpuset-mems=0;

– **Ограничение ввода-вывода (I/O) для дисков:**

– ограничение скорости чтения с диска: --blkio-read-bps-device /dev/sda:10mb;

– ограничение скорости записи: --blkio-write-bps-device /dev/sda:5mb;

– ограничение количества операций ввода-вывода (IOPS): --blkio-read-iops-device /dev/sda:100 --blkio-write-iops-device /dev/sda:50

– **Ограничения по PID и процессам:**

– максимальное количество процессов в контейнере: --pids-limit=200

10.2.4 Интеграция Podman с systemd

Podman поддерживает генерацию unit-файлов systemd для автоматического запуска контейнеров и подов. Управление контейнерами средствами systemd обеспечивает предсказуемый запуск сервисов, ведение журналов и использование стандартных механизмов контроля ресурсов ОС.

10.2.4.1 Генерация unit-файлов

Для существующего контейнера можно сформировать unit-файл systemd:

```
podman generate systemd --new --files --name <имя_контейнера>
```

где параметры определяют:

– --new — что контейнер создаётся заново при каждом запуске systemd;
– --files — сохранение unit-файлов в текущий каталог;
– --name — имя контейнера.

Пример:

```
podman run -d --name web -p 8080:80 nginx
podman generate systemd --new --files --name web
```

10.2.4.2 Размещение и активация unit-файлов

Для активации unit-файлов в Rootless-режиме файл должен быть размещён в каталоге ~/.config/systemd/user/.

После переноса файла необходимо обновить конфигурацию systemd:

```
systemctl --user daemon-reload
systemctl --user enable --now container-web.service
```

Для автозапуска требуется активировать параметр linger, с помощью которого контейнеры запускаются без активной пользовательской сессии.:

```
sudo loginctl enable-linger <user>
```

Для сертифицированной ОС unit-файлы должны иметь права "600", а также иметь заданные лимиты ресурсов (MemoryMax, TasksMax, CPUQuota).

Пример параметров для unit-файла:

```
[Unit]
Description=Podman Container: web
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
Restart=always
ExecStart=/usr/bin/podman run --rm --name web -p 8080:80
nginx
ExecStop=/usr/bin/podman stop -t 10 web

#ограничения (рекомендовано для сертифицированной версии ОС)
MemoryMax=512M
CPUQuota=50%
TasksMax=100

[Install]
WantedBy=default.target
```

10.2.4.3 Управление контейнерами как службами systemd

Формат команд для rootless-режима:

```
systemctl --user start container-web.service
systemctl --user status container-web.service
systemctl --user stop container-web.service
```

Формат команд для rootful-режима:

```
sudo systemctl start container-web.service
sudo systemctl status container-web.service
sudo systemctl stop container-web.service
```

Проверка журналов:

```
journalctl --user -u container-web.service
```

10.2.4.4 Управление подами через systemd

Podman может генерировать unit-файлы не только для отдельных контейнеров, но и для подов.

Создание pod:

```
podman pod create --name app-pod -p 9000:9000
```

Добавление контейнера в pod:

```
podman run -d --pod app-pod --name backend my-backend-image
```

Генерация единых unit-файлов:

```
podman generate systemd --new --files --name app-pod
```

В результате будет создано два файла:

– pod-app-pod.service — управляет жизненным циклом pod;
– container-backend.service — управляет конкретным контейнером.

10.3 Kubernetes

10.3.1 Общие сведения

Kubernetes – это система с открытым исходным кодом для развертывания, масштабирования и управления контейнезированными приложениями.

Kubernetes по сути является не просто системой оркестрации. Технически, оркестрация – это про выполнение определенного рабочего процесса: сначала нужно сделать A, затем B, затем C. Kubernetes же устраняет прямую необходимость в этом. В нем есть процессы управления, по факту независимые и компонуемые. Главная задача процессов управления – перевести текущее состояние к нужному состоянию. В этом случае неважно, какой будет маршрут от A до C, что исключает централизованный контроль. Благодаря этому Kubernetes проще в использовании, мощная, надежная, а также устойчивая и расширяемая.

Контейнеры позволяют поделить приложения на более мелкие части с четким разделением задач. Уровень абстракции, предоставляемый для отдельного образа контейнера, позволяет понять, как строятся распределенные приложения. Такой модульный подход дает возможность быстрее осуществлять разработку с помощью небольших и более целенаправленных групп, каждая из которых отвечает за определенные контейнеры. Это также позволяет изолировать зависимости и шире использовать компоненты меньшего размера. Сделать это только с помощью контейнеров не получится. А вот в Kubernetes это можно достичь с помощью Pods (подов). Pod (под) – это группа из одного или более контейнеров с общим хранилищем/сетевыми ресурсами и спецификацией

запуска контейнеров или отдельный инстанс (экземпляр объекта) приложения. Размещая контейнеры таким образом, Kubernetes устраняет соблазн втиснуть слишком много функций в один образ контейнера. Концепция Service (Сервисы) в Kubernetes используется для группирования нескольких подов, которые выполняют те же функции. Сервисы легко настраиваются для таких целей, как обнаружение, горизонтальное масштабирование и балансировка нагрузки. Kubernetes согласно официальной документации так же сможет предоставить:

– Мониторинг сервисов и распределение нагрузки, используя имя DNS или собственный IP-адрес, Kubernetes может обнаружить контейнер. При высоком трафике в нем Kubernetes сбалансирует нагрузку и распределит сетевой трафик так, что развертывание будет стабильным.

– Система хранения по выбору (например, локальное хранилище, провайдеры общедоступного облака и многое другое) может быть автоматически смонтирована с помощью оркестрации хранилища Kubernetes.

– Автоматическое развертывание и откаты. Kubernetes через описание желаемого состояния развернутых контейнеров (манифесты, которые пишутся на YAML) может изменить фактическое состояние на желаемое. То есть создание новых контейнеров для развертывания, удаления существующих контейнеров и распределения всех их ресурсов в новый контейнер в Kubernetes можно автоматизировать.

– Автоматическое распределение нагрузки. Kubernetes сам размещает контейнеры на узлах так, чтобы наиболее эффективно использовать ресурсы. Администратору остается только указать сколько ЦП и ОЗУ требуется каждому контейнеру и предоставить кластер узлов, на котором будут запущены контейнеры.

– Самоконтроль. Если в работе контейнеров что-то пошло не так, то Kubernetes сам перезапускает, заменяет и завершает работу контейнеров, которые не проходят проверку работоспособности.

– Управление конфиденциальной информацией и конфигурацией. Пароли, OAuth-токены и SSH-ключи могут храниться и управляться Kubernetes без изменений образов контейнеров и без раскрытия конфиденциальной информации в конфигурации стека.

10.3.2 Установка kubernates master-node

Установка kubernates master-node проводится следующим образом:

а) Установка всех зависимостей kubernates:

```
sudo dnf install -y kubelet kubeadm kubectl docker
```

б) Запуск и добавление в автозагрузку сервисов:

```
sudo systemctl enable docker.service
```

```
sudo systemctl enable containerd.service
sudo systemctl start docker.service
sudo systemctl start containerd.service
sudo systemctl enable kubelet
sudo systemctl start kubelet
```

в) Добавление узлов в hosts (master, worker):

```
sudo hostnamectl set-hostname master-node
vi /etc/hosts
10.45.4.58 master-node
10.45.4.59 worker-node
```

г) Отключение swap:

```
sudo sed -i '/swap/d' /etc/fstab
sudo swapoff -a
```

д) Инициализация kubernates:

```
kubeadm init
```

е) Выдача установщиком токена подключения к master node:

```
kubeadm join 10.45.4.58:6443 --token ncb63c.pugf9xehkjakptqp
--discovery-token-ca-cert-hash
sha256:73cb40755a49b40f2a724984392f32fd7ff607a437da524619d4aba43
a2c31b6
```

ж) Создание каталогов, необходимых для kubernates:

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
export kubever=$(kubectl version | base64 | tr -d '\n'
```

з) Добавление сети для работоспособности kubernates:

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-
version=$kubever"
```

и) Перезагрузка kubernates:

```
sudo systemctl restart kubelet
```

10.3.3 Установка kubernates worker-nodes (pods-контейнеры)

Установка kubernates worker-nodes выполняется следующими шагами:

а) Установка docker:

```
sudo dnf install docker
```

б) Запуск и добавление в автозагрузку сервисов docker:

```
sudo systemctl start docker
sudo systemctl enable docker
```

в) Установка всех зависимостей kubernates:

```
sudo dnf install -y kubelet kubeadm kubectl
```

г) Добавление автозагрузку kubernates:

```
sudo systemctl enable kubelet
```

д) Изменение имени виртуального узла:

```
sudo hostnamectl set-hostname worker-node
```

е) Добавление узлов в hosts:

```
vi /etc/hosts
10.45.4.58 master-node
10.45.4.59 worker-node
```

ж) Отключение swap:

```
sudo sed -i '/swap/d' /etc/fstab
sudo swapoff -a
```

з) Присоединение worker-node к master (токен – из предыдущего этапа):

```
kubeadm join 10.45.4.58:6443 --token ncb63c.pugf9xehkjakptqp
--discovery-token-ca-cert-hash
sha256:73cb40755a49b40f2a724984392f32fd7ff607a437da524619d4aba43
a2c31b6
```

и) Перезапуск сервиса:

```
sudo systemctl restart kubelet.service
```

к) Создание deployment-плана:

```
vi deploy.yml
apiVersion : apps/v1 #версия api
kind: Deployment #Принципал deployment
metadata: #Метаданные
 name: my-web-deployment
 labels:
 app : my-k8s-application
```

```
spec: #Описание контейнера
 selector:
 matchLabels:
 project: kgb #Проект
 template: #Шаблон для контейнера
 metadata:
 labels:
 project: kgb
 spec:
 containers: #Контейнер
 - name : kgb-web #Имя контейнера
 image: Debian #Имидж на котором будет основываться контейнер
 ports: #Порт который нужно пробросить в хост машину
 - containerPort: 80
```

л) Применение deployment-плана:

```
#kubectl apply –f deploy.yml
```

11 СБОР ОТЧЕТОВ О ПРОИЗВОДИТЕЛЬНОСТИ

11.1 Общие сведения о Zabbix

Для эффективного мониторинга Системы рекомендуется использовать службу Zabbix. Это мощное решение для мониторинга, которое способствует обеспечению стабильной работы Системы и своевременному обнаружению возможных проблем.

Пакеты Zabbix должны быть установлены администратором Системы самостоятельно, так как они не входят в сборку ОС.

Службы Zabbix являются универсальными инструментами по сбору отчетов и могут адаптироваться под различные масштабы и спецификации Систем: от небольших до крупных сред, позволяя контролировать множество устройств и сервисов.

С помощью Zabbix возможно выполнение следующего функционала в ОС:

– Отслеживание разнообразных метрик Системы, таких как использование процессора, память, дисковое пространство, сетевую активность и много других параметров, что позволяет получать полное представление о состоянии Системы;

– Настройка своих триггеров и уведомлений, что обеспечивает возможность мгновенной реакции на возникающие инциденты. Например, настройка уведомления о превышении лимитов загрузки CPU или исчерпании дискового пространства;

– Предоставление данных и отчетности по производительности Системы в графическом режиме.

Система Zabbix также предоставляет RESTful API для интеграции с другими системами или создания собственных автоматизированных процессов.

Компоненты службы Zabbix:

– Zabbix Server – основной компонент, который отвечает за сбор данных, хранение информации и обработку триггеров.

– Zabbix Agent – устанавливается на целевых системах и собирает данные о производительности и состоянии системных ресурсов, отправляя их на сервер.

– Zabbix Frontend – веб-интерфейс через веб-сервер Apache HTTPD, который позволяет администраторам и пользователям взаимодействовать с системой, осуществляя настройку, просмотр статистики и управление уведомлениями.

11.2 Установка и настройка

11.2.1 Установка компонентов Zabbix

Процесс установки и настройки Zabbix:

а) Установка сервера и агента:

1) Сначала необходимо добавить пакеты с Zabbix в Систему:

```
sudo dnf install zabbix-server zabbix-server-mysql zabbix-web zabbix-web-mysql mariadb locales-ru
```

Это установит сервер Zabbix, веб-интерфейс Zabbix, компоненты для использования MySQL в качестве базы данных и саму базу данных – MariaDB (форк MySQL).

2) Установка имени хоста (это делать не обязательно, но рекомендуется), например:

```
sudo hostnamectl set-hostname\ zabbix.infrastructure.имя_хоста.ru
```

б) Настройка базы данных:

1) После установки MariaDB необходимо настроить базу данных для Zabbix. Для этого необходимо подключиться к MariaDB и создать базу данных и пользователя:

```
sudo mysql -uroot -p
```

2) Ввести SQL-запрос для создания базы данных и пользователя:

```
create database zabbix character set utf8 collate utf8_bin;
create user zabbix@localhost identified by 'пароль';
grant all privileges on zabbix.* to zabbix@localhost;
quit;
```

3) Заменить параметр "пароль" на пароль, который будет использоваться для БД.

в) Импорт начальных схем и данных в базу данных Zabbix:

```
zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix -p zabbix
```

г) Настройка Zabbix Server:

1) Настройка файла конфигурации Zabbix Server, с указанием параметров подключения к базе данных. Для этого необходимо открыть и отредактировать файл /etc/zabbix/zabbix_server.conf:

```
sudo nano /etc/zabbix/zabbix_server.conf
```

2) Найти и изменить строки следующим образом:

```
DBName=zabbix
DBUser=zabbix
DBPassword=password
```

где необходимо заменить параметр password на пароль, который был задан ранее для пользователя базы данных Zabbix.

д) Запуск службы Zabbix Server и Zabbix Agent и настройка их на автоматический запуск при старте Системы:

```
sudo systemctl restart zabbix-server zabbix-agent apache2
sudo systemctl enable zabbix-server zabbix-agent apache2
```

11.2.2 Настройка веб-интерфейса

Шаги по настройке веб-интерфейса Zabbix, которые необходимо выполнить:

а) Настроить PHP:

Открыть файл /etc/zabbix/apache.conf и настроить параметры PHP для Zabbix:

```
sudo nano /etc/zabbix/apache.conf
```

Обновить следующие значения параметров:

```
php_value date.timezone Europe/Moscow
```

Убедиться, что установлен корректный часовой пояс, соответствующий нужному региону.

б) Перезапустить веб-сервер Apache, чтобы применить изменения:

```
sudo systemctl restart apache2
```

в) Первоначальная настройка через веб-интерфейс. Открыть веб-браузер и перейти по адресу http://your_server_ip/zabbix, где вместо your_server_ip нужно указать IP-адрес пользовательского сервера. Далее нужно следовать инструкциям на экране, чтобы завершить установку.

В ходе настройки потребуется ввести имя базы данных, пользователя и пароль, которые были настроены ранее.

11.2.3 Установка Zabbix Agent на целевых машинах

Для подключения целевых машин к Zabbix Agent необходимо:

а) Установить Zabbix Agent на целевые машины:

```
sudo apt-get install zabbix-agent
```

б) Настройка Zabbix Agent, для этого открыть на редактирование файл конфигурации /etc/zabbix/zabbix_agentd.conf:

```
sudo nano /etc/zabbix/zabbix_agentd.conf
```

Найдите и измените следующие строки:

```
Server=your_zabbix_server_ip
ServerActive=your_zabbix_server_ip
Hostname=your_target_system_hostname
```

Заменить параметр your_zabbix_server_ip на IP-адрес вашего сервера Zabbix и your_target_system_hostname на имя хостовой машины.

в) Запуск Zabbix Agent и добавление его на автоматический запуск при старте целевой системы:

```
sudo systemctl restart zabbix-agent
sudo systemctl enable zabbix-agent
```

11.2.4 Завершение настройки

Далее можно вернуться в веб-интерфейс Zabbix и добавить новые хосты для мониторинга. В разделе "Configuration → Hosts" добавляются целевые системы, на которых установлен Zabbix Agent, и настраиваются необходимые параметры мониторинга.

12 РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ

12.1 Системы управления базами данных

Система управления базами данных (далее – СУБД) – это комплекс программно-языковых средств, позволяющих создавать базы данных и управлять данными.

12.1.1 Доступные СУБД

В ОС доступны для использования три СУБД:

– MySQL – реляционная клиент-серверная СУБД; для проектов небольшого и среднего размера; легкая, гибкая и довольно простая в использовании; бесплатная; хорошо подходит для обучения и веб-проектов;

– SQLITE – небольшая и легкая встраиваемая СУБД, которая обычно применяется в локальных проектах;

– PostgreSQL – объектно-реляционная СУБД клиент-серверного типа, которую иногда называют бесплатным аналогом Oracle; масштабируемая; для высоконагруженных проектов; содержит огромное количество функций и распространяется бесплатно.

12.1.2 PostgreSQL

Свободная объектно-реляционная PostgresSQL присутствует в репозиториях ОС. Для установки используют следующую команду:

```
sudo dnf install postgresql-server
```

PostgreSQL использует концепцию ролей для выполнения аутентификации и авторизации клиента. В некоторых аспектах они напоминают обычные учетные записи в ОС, но PostgresSQL не делает различий между пользователями и группами и предпочитает использовать более гибкий термин – роль.

После установки PostgresSQL настроена на использование аутентификации ident, это означает выполнение привязки ролей PostgresSQL к системной учетной записи ОС – postgres. Если роль существует внутри PostgresSQL, пользователь ОС с тем же именем может выполнить вход в качестве этой роли.

Чтобы использовать PostgreSQL, можно войти в эту учетную запись. Но сначала нужно установить для нее свой пароль:

```
passwd postgres
```

Чтобы PostgresSQL запускалась с загрузкой ОС, нужно включить сервис:

```
sudo systemctl enable postgresql12.service
```

Чтобы запустить PostgreSQL, необходимо запустить соответствующий сервис:

```
sudo systemctl start postgresql12.service
```

Для проверки факта запуска PostgreSQL и его версии нужно выполнить команду:

```
sudo -u postgres psql -c "SELECT version();"
```

В результате запуска можно получить доступ к командной строке PostgreSQL, а затем свободно взаимодействовать с СУБД с помощью команды:

```
sudo -u postgres psql
```

Например, с помощью команд можно:

– Создать роль testuser:

```
postgres=# create user testuser;
```

– Создать пароль для роли testuser:

```
postgres=# \password testuser
```

– Создать базу данных testdb:

```
postgres=# create database testdb;
```

– Предоставить разрешения пользователю testuser в базе данных testdb:

```
postgres=# grant connect, create on database testdb to testuser;
postgres=# grant all privileges on database testdb to testuser;
```

– Просмотреть базы данных:

```
postgres=# \l
```

– Просмотреть роли:

```
postgres=# \du
```

– Выйти из консоли psql:

```
postgres=# \q
```

12.1.3 MySQL

MySQL – это одна из самых популярных систем управления реляционными базами данных, основанная на языке запросов SQL (Structured Query Language).

MySQL отличается высокой производительностью, надежностью и широким применением в серверных ОС. Она поддерживает модель клиент-сервер, что позволяет эффективно управлять данными в многопользовательских системах.

Основное преимущество MySQL заключается в поддержке многопоточности, позволяющей работать с большими объемами данных и обрабатывать несколько запросов одновременно. MySQL используется для веб-сайтов, облачных платформ, а также для крупных корпоративных приложений благодаря совместимости с различными ОС и интеграции с популярными веб-серверами, такими как Apache.

Установка СУБД MySQL описана в п.6.4.1.3.

12.1.4 SQLITE

SQLite – это легковесная встраиваемая система управления базами данных, которая не требует отдельного сервера для работы и хранит все данные в одном файле. Она широко применяется для локальных баз данных или приложений, которые не нуждаются в сложной многопользовательской архитектуре.

SQLite работает с минимальными ресурсами и подходит для приложений с низкой нагрузкой на БД, таких как десктопные программы, мобильные приложения, а также для разработки и тестирования. SQLite является ACID-совместимой (атомарность, согласованность, изолированность, долговечность), поддерживает основные SQL-запросы и транзакции. Однако, в отличие от MySQL или PostgreSQL, она не предназначена для масштабных многопользовательских систем, так как не поддерживает полноценную многопоточность и функции клиент-серверной архитектуры.

Для установки SQLite в ОС нужно использовать команду:

```
sudo dnf install sqlite
```

Для получения справки по командам и функциям, связанным с SQLite, можно использовать встроенную в утилиту команду:

```
sqlite3 --help
```

Также можно обратиться к официальной документации SQLite или прочитать справку в формате man-страницы:

```
man sqlite3
```

Эти команды предоставят подробное руководство по использованию SQLite, включая основные команды для работы с базами данных.

12.2 Локальная виртуализация с использованием qemoo

Утилита `qemoo` представляет собой обёртку для запуска виртуальных машин с использованием эмулятора QEMU. Скрипт предназначен для упрощения командной строки QEMU и автоматизации типовых операций, таких как:

– запуск ISO-образов и виртуальных дисков;
– автоматическое создание дисков в формате qcow2;
– эмуляция загрузки в режиме EFI;
– проброс физических устройств и каталогов;
– работа через systemd и поддержка демон-режима SPICE;
– настройка сетевых интерфейсов виртуальных машин;
– управление конфигурацией виртуальных машин через конфигурационные файлы.

Утилита входит в состав дистрибутива ОС и предназначена для использования администраторами при отладке, тестировании, развёртывании и сопровождении виртуальных инсталляций ОС.

Для запуска виртуальной машины с использованием `qemoo` достаточно передать путь к ISO-образу, блочному устройству или файлу виртуального диска. Примеры:

```
qemoo Rosa.iso
qemoo /dev/sda
qemoo ./Rosa.qcow2
```

При необходимости загрузки в режиме EFI используется ключ -e:

```
qemoo -e Rosa.qcow2
```

Дополнительные ключи и параметры позволяют расширить функциональность скрипта и адаптировать его под конкретные задачи виртуализации. Все параметры передаются в одной командной строке, а конфигурация может задаваться как явно, так и через конфигурационные файлы, размещённые локально или глобально.

12.2.1 Запуск виртуальных машин с помощью qemoo

Для запуска виртуальных машин с использованием `qemoo` необходимо передать в качестве аргумента путь к ISO-образу, блочному устройству или образу виртуального диска в формате .qcow2. Утилита самостоятельно определяет параметры запуска на основе переданного файла и применяет настройки по умолчанию или из конфигурационных файлов.

Примеры запуска:

```
qemoo Rosa.iso
qemoo /dev/sda
qemoo ./Rosa.qcow2
```

12.2.1.1 Загрузка в режиме EFI

При необходимости загрузки в режиме EFI используется ключ -e:

```
qemoo -e Rosa.qcow2
```

Также может быть применена комбинированная команда:

```
qemoo -i -e /dev/sdb
```

В этом случае будет произведена установка с ISO в EFI-режиме на указанный диск или образ.

12.2.1.2 Добавление дополнительных устройств и параметров

Для подключения дополнительных устройств (например, второго диска или USB) используется ключ -a. Пример запуска ISO-образа с подключением второго устройства:

```
qemoo Rosa.iso -a /dev/sdb
```

Параметры, поддерживаемые эмулятором QEMU, могут передаваться напрямую после двойного дефиса "--". Например, передача параметра количества виртуальных процессоров:

```
qemoo -e Rosa.iso -- -smp 4
```

12.2.1.3 Загрузка в режиме, имитирующем запись ISO-образа на носитель

Скрипт поддерживает режим, в котором ISO-образ загружается так, как если бы он был записан побайтово на флеш-накопитель. Для этого используется ключ -l:

```
qemoo -l Rosa.iso
qemoo -l -e Rosa.iso
```

12.2.1.4 Проброс USB-устройств

Для проброса физических USB в виртуальную машину используется ключ -L, за которым следует путь к устройству:

```
qemoo -L /dev/sdb
```

Такая схема может быть использована, например, для загрузки виртуальной машины с USB-модема или токена, содержащего ОС.

12.2.2 Установка ОС на виртуальный диск

Скрипт qemoo поддерживает автоматизированную установку ОС с ISO-образа на виртуальный диск в формате .qcow2. Для этого используется ключ -i.

При запуске с данным ключом в текущем каталоге создаётся файл виртуального диска, который автоматически подключается к виртуальной машине при запуске.

Примеры установки:

```
qemoo -i Rosa.iso
```

В режиме EFI:

```
qemoo -i -e Rosa.iso
```

Также возможно указание целевого устройства (например, реального диска /dev/sdb):

```
qemoo -i -e /dev/sdb
```

После завершения установки возможно использование ранее созданного .qcow2-образа для загрузки, передав его в qemoo как аргумент:

```
qemoo _qemoo1_ROS.qcow2
```

Имена создаваемых образов формируются автоматически, но могут быть заданы и вручную.

Также поддерживается установка с ISO-образа на физический диск. В этом случае используется ключ -a, указывающий дополнительное устройство для записи:

```
qemoo Rosa.iso -a /dev/sdb
```

Во всех режимах установки применяется типовая конфигурация QEMU, достаточная для корректной загрузки и установки ОС РОСА "ХРОМ".

12.2.3 Сетевые режимы и проброс устройств

При запуске виртуальных машин с использованием `qemoo` автоматически настраивается сетевое подключение, обеспечивающее взаимодействие гостевой системы с хостом и другими машинами в сети.

12.2.3.1 Автоматическое подключение к подсети virbr0

Если на хост-системе установлена и запущена служба libvirtd, виртуальные машины, запущенные с помощью qemoo, автоматически получают IP-адрес из подсети 192.168.122.0/24, связанной с виртуальным интерфейсом virbr0.

Такая конфигурация обеспечивает полноценную сетевую связность и доступ к интернету через NAT. В случае отсутствия интерфейса virbr0 (например, если libvirtd не установлен или не активен) для каждой виртуальной машины создаётся собственная NAT-сеть, что может ограничивать взаимодействие между гостевыми системами и внешней средой.

12.2.3.2 Настройка собственного сетевого моста

Для получения большей гибкости возможно создание и использование собственного сетевого моста с именем qemoobr0. При наличии такого моста утилита qemoo игнорирует интерфейс virbr0 и использует qemoobr0 для подключения виртуальных машин.

Настройка моста производится средствами Системы. Конкретные действия зависят от используемой сетевой конфигурации хоста.

12.2.3.3 Проброс каталогов в гостевую систему

При запуске виртуальной машины с помощью qemoo может быть автоматически проброшен каталог с хоста внутрь гостевой системы. По умолчанию пробрасывается текущий каталог (./), из которого выполняется запуск. При старте виртуальной машины в консоль выводится инструкция по подключению каталога внутри гостевой ОС на базе Linux.

12.2.3.4 Проброс USB-устройств

Для проброса USB-устройства в виртуальную машину используется ключ -L. Указывается путь к устройству:

```
qemoo -L /dev/sdb
```

Этот режим может быть использован, например, для загрузки с 4G-модема с SD-картой, токена с предустановленной Системой или иных внешних накопителей.

12.2.4 Работа в демон-режиме (SPICE)

Утилита qemoo поддерживает запуск виртуальной машины в виде фонового демона с выводом графического интерфейса через протокол SPICE. Это

позволяет подключаться к экрану гостевой ОС с помощью внешних клиентов, таких как virt-viewer или Remmina.

12.2.4.1 Запуск в SPICE-режиме

Для запуска виртуальной машины в демон-режиме используется ключ -d. При этом `qemoo` автоматически выделяет свободный порт и сообщает его в вывод консоли:

```
qemoo -d Rosa.iso
```

Вывод будет содержать номер порта, на который можно подключиться из клиента SPICE. Подключение выполняется вручную с указанием IP-адрес хоста и полученного порта.

12.2.4.2 Управление виртуальной машиной через systemd

При запуске виртуальной машины с ключом -i (установка ОС) `qemoo` создаёт рядом с образом файл конфигурации с расширением .conf. Этот файл содержит параметры запуска и может использоваться для интеграции с systemd.

Пример содержимого конфигурационного файла:

```
ACTION=run
RAM="auto"
ADD=""
EFI="-bios /usr/share/OVMF/OVMF_CODE.fd"
PORT=""
REDIRUSB=""
LOSETUP=""
SPICE=""
SHARE="/home/user/Downloads"
QEMOOADD=""
```

Для запуска виртуальной машины от имени пользователя root через systemd применяется следующая команда:

```
sudo systemctl start qemoo@$(systemd-escape\
/путь/к/образ.qcow2)
```

Проверка статуса виртуальной машины:

```
systemctl status qemoo@$(systemd-escape /путь/к/образ.qcow2)
```

Для включения автозапуска виртуальной машины при старте хост-системы:

```
    sudo systemctl enable qemoo@$(systemd-escape\
/путь/к/образ.qcow2)
```

Отключение автозапуска:

```
    sudo systemctl disable qemoo@$(systemd-escape\
/путь/к/образ.qcow2)
```

Запуск от имени текущего пользователя с использованием user-юнитов:

```
    sudo systemctl --user enable qemoo@$(systemd-escape\
/путь/к/образ.qcow2)
```

Команда `systemd-escape` используется для корректного экранирования символов "/" в путях файловой системы при создании unit-имен.

12.2.5 Конфигурация и приоритет параметров

Скрипт qemoo поддерживает использование конфигурационных файлов, в которых могут задаваться параметры запуска виртуальных машин. Эти параметры могут быть как общими для всех запусков, так и специфичными для отдельных образов. Это позволяет централизованно управлять поведением qemoo без необходимости передавать параметры вручную в командной строке при каждом запуске.

При запуске qemoo учитываются конфигурационные файлы в следующем порядке приоритета (от меньшего к большему):

а) глобальный файл – /etc/qemoo.cfg. Используется по умолчанию, если переменная окружения $QEMOOCFG не задана;

б) локальный файл в текущем каталоге – ./qemoo.cfg;

в) индивидуальный конфигурационный файл для образа: если виртуальная машина запускается из файла образа example.qcow2, то приоритет будет иметь файл example.qcow2.conf, расположенный рядом с ним;

г) конфигурационный файл, явно указанный в параметре: при запуске с опцией --config – /путь/к/файлу.cfg.

Если в конфигурационных файлах заданы одинаковые параметры, приоритет имеет значение из файла с более высоким уровнем.

12.2.5.1 Типовые параметры конфигурации

В конфигурационном файле могут быть заданы следующие переменные:

– ACTION – тип действия (run, install и др.);
– RAM – объём оперативной памяти, выделяемой виртуальной машине;

– ADD – путь к дополнительному устройству (например, дополнительный диск);

– EFI – параметры для запуска в режиме EFI;

– PORT – порт для SPICE-подключения;

– SPICE – активация вывода видео через SPICE (yes / no);

– REDIRUSB – проброс USB-устройств;

– LOSETUP – использование losetup при работе с образом;

– SHARE – каталог, пробрасываемый в гостевую ОС;

– QEMOOADD – дополнительные параметры командной строки для qemu-system-*.

Пример конфигурационного файла:

```
ACTION=run
RAM=2048
ADD=/dev/sdb
EFI="-bios /usr/share/OVMF/OVMF_CODE.fd"
SPICE=yes
PORT=6001
SHARE=/home/user/Projects
```

13 СОЗДАНИЕ СОБСТВЕННОГО РЕПОЗИТОРИЯ ПАКЕТОВ

Репозиторий пакетов представляет собой хранилище RPM-файлов, дополненное служебными файлами метаданных, обеспечивающими установку и обновление пакетов с использованием менеджера пакетов dnf. Подробнее о работе с dnf см. раздел 3.6.2.

Создание собственного репозитория может быть необходимо для решения следующих задач:

– централизованное размещение сторонних пакетов, используемых в организации, для их установки и обновления из единого источника;

– создание зеркала (точной копии) официальных репозиториев РОСА в пределах локальной сети предприятия.

Важно – Все команды, требующие прав суперпользователя, должны выполняться с использованием sudo. В остальных случаях выполнение команд производится от имени непривилегированного пользователя.

13.1 Зеркалирование репозиториев

Зеркалирование репозитория заключается в создании и регулярном обновлении локальной копии репозиториев, размещённых на официальных серверах РОСА. Такое зеркало позволяет сократить внешнее сетевое взаимодействие и ускорить установку и обновление пакетов внутри корпоративной сети.

Для зеркалирования рекомендуется использовать утилиту rsync. Инструкция по созданию зеркала с использованием rsync:

а) Если утилита rsync не установлена, необходимо выполнить установку командой:

```
sudo dnf install rsync
```

б) Создание каталогов для хранения копий репозиториев:

```
sudo mkdir -p /srv/mirror/x86_64 /srv/mirror/i686
```

в) Запуск синхронизации с официальным сервером:

```
sudo rsync \
  -av --progress \
  --exclude SRPMS \
  --exclude 'debug_*' \
  --exclude ___REMOVED \
  --exclude '*-rpm-backup' \
```

```
rsync://mirror.rosalab.ru/rosa/rosa2021.1/repository/{x86_64,i686} /srv/mirror/
```

В приведённой команде синхронизируются репозитории для архитектур x86_64 и i686.

Каталоги, содержащие пакеты с отладочной информацией (debug_*) и SRPM-исходники (SRPMS), исключаются из копирования для экономии дискового пространства.

Регулярное выполнение задачи синхронизации рекомендуется автоматизировать с использованием планировщика systemd timer. Подробная инструкция по созданию планируемых задач приведена в разделе 3.4.2.1.

13.2 Создание собственного репозитория

Если необходимо использовать сторонние RPM-пакеты, можно организовать собственный репозиторий для централизованного хранения и распространения этих пакетов.

Для создания репозитория необходимо выполнить следующие действия:

а) установить утилиту для формирования метаданных репозитория:

```
sudo dnf install createrepo_c
```

б) создать каталог для хранения пакетов:

```
sudo mkdir -p /srv/repo/x86_64
```

в) поместить все необходимые RPM-пакеты в созданный каталог /srv/repo/x86_64.

г) создать или обновить метаданные репозитория:

```
sudo createrepo_c /srv/repo/x86_64
```

д) для обновления репозитория после добавления или удаления пакетов необходимо повторно выполнить команду на шаге г);

е) для проверки содержимого репозитория используется следующая команда:

```
dnf --repofrompath=local,/srv/repo/x86_64 --disablerepo='*' --enablerepo=local list
```

13.3 Подключение локального репозитория

Для локального подключения репозитория необходимо создать конфигурационный файл в каталоге /etc/yum.repos.d/. В файле указывается путь к репозиторию на локальной Системе.

Пример файла myrepo.repo:

```
[myrepo]
name=My Local Repository
baseurl=file:///path/to/my/repo
enabled=1
gpgcheck=0
```

где /path/to/my/repo – путь к каталогу, содержащему RPM-пакеты и созданные метаданные.

После создания конфигурационного файла пакеты из локального репозитория становятся доступными для установки с помощью dnf наравне с другими подключёнными репозиториями.

13.4 Доступ к репозиторию по сети

Организовать сетевой доступ к локальному репозиторию пакетов можно с использованием HTTP или NFS.

13.4.1 Доступ по HTTP

Для предоставления доступа по HTTP-протоколу достаточно использовать любой веб-сервер, способный отдавать статические файлы. В качестве примера рассмотрена настройка веб-сервера angie (об этом подробнее см. раздел 6.4.3).

Для настройки HTTP-сервера нужно выполнить следующие шаги:

а) Установить веб-сервер:

```
sudo dnf install angie
```

б) Открыть основной конфигурационный файл на редактирование:

```
sudo nano /etc/angie/angie.conf
```

в) Привести файл к необходимому виду. Ниже приведён минимальный пример:

```
worker_processes  1;

events {
```

```
        worker_connections  1024;
    }

    http {
        include       mime.types;
        default_type  application/octet-stream;
        sendfile        on;
        keepalive_timeout  65;

        server {
            listen       80;
            server_name  localhost;

            location / {
                root   /srv;
                autoindex on;
                index  index.html index.htm;
            }

            error_page   500 502 503 504  /50x.html;
            location = /50x.html {
                root   html;
            }
        }
    }
```

Для выхода и сохранения изменений в редакторе nano использовать комбинации клавиш: `Ctrl+X`, затем `Enter` и `Ctrl+O`.

г) Запустить веб-сервер и добавить его в автозапуск:

```
sudo systemctl enable --now angie
```

д) Проверить доступ к репозиторию, открыв в браузере адрес http://<IP-адрес_сервера>:80. Должно отобразиться содержимое каталога /srv.

Примечание – Если требуется только временный или тестовый доступ, можно использовать встроенный модуль Python:

```
python3 -m http.server
```

По умолчанию сервер запускается на порту 8000. В конфигурационном файле репозитория указать адрес в формате:

```
baseurl=http://192.168.1.100:8000
```

Для настройки HTTP-клиента необходимо:

а) В файлах конфигурации /etc/yum.repos.d/*.repo указать адрес собственного сервера. Пример:

Было:

```
baseurl=http://mirror.rosalab.ru/rosa/rosa$releasever/repository/x86_64/main/release/
```

Стало:

```
baseurl=http://192.168.1.100/repo/x86_64/rosa$releasever/repository/x86_64/main/release/
```

При необходимости можно указать несколько репозиториев через запятую — в случае недоступности первого будет использован следующий.

б) Проверить работу подключённого репозитория:

```
sudo dnf makecache
```

13.4.2 Доступ по NFS

Для организации доступа по сети с использованием NFS необходимо настроить NFS-сервер и клиент.

Для настройки NFS-сервера необходимо:

а) Установить утилиты NFS:

```
sudo dnf install nfs-utils
```

б) Открыть файл конфигурации сервера:

```
sudo nano /etc/exports
```

в) Добавить строку с настройками общего доступа. Пример:

```
/srv 192.168.1.1/255.255.255.0(ro,insecure,nohide,all_squash,anonuid\=1000,anongid=1000,no_subtree_check)
```

В приведённом примере доступ предоставляется только на чтение для сети с адресами от 192.168.1.1 до 192.168.1.255.

г) Запустить сервер и добавить его в автозапуск:

```
sudo systemctl enable --now nfs-server
```

Для настройки NFS-клиента необходимо:

а) Создать каталог для монтирования:

```
sudo mkdir -p /mnt/nfs
```

б) Открыть файл /etc/fstab для редактирования:

```
sudo nano /etc/fstab
```

в) Добавить строку монтирования NFS-ресурса:

```
192.168.1.100:/srv /mnt/nfs nfs4 auto,_netdev,bg,nofail
```

где 192.168.1.100 – адрес NFS-сервера.

г) Выполнить монтирование для проверки:

```
sudo mount /mnt/nfs
```

д) Указать путь к локальному репозиторию в файле /etc/yum.repos.d/*.repo:

Заменить содержимое строки:

```
baseurl=http://mirror.rosalab.ru/rosa/rosa$releasever/repository/x86_64/main/release/
```

на следующее содержимое:

```
baseurl=file:///mnt/nfs/srv/repo/x86_64/rosa$releasever/repository/x86_64/main/release/
```

е) Проверить работу подключённого репозитория:

```
sudo dnf makecache
```

Вывод должен представлять собой список репозиториев, подключённых в Системе и успешно обработанных пакетным менеджером dnf.

ПРИЛОЖЕНИЕ А. СПИСОК ДОСТУПНЫХ КОНСОЛЬНЫХ КОМАНД

Команда	Описание
A	
a2p	Конвертирует awk-скрипт в Perl-скрипт
a2ps	Форматирует текстовый файл для печати на Postscript-принтере
access	Проверяет, имеет ли вызывающая программа доступ к указанному файлу
accton	Используется администратором для включения/выключения ведения журнала действий пользователей (лог-файла). Регистрация активности пользователей сохраняется в заданном текстовом файле
aclocal	Автоматическое создание файлов aclocal.m4 на основе содержимого файлов configure.in
acpi_available	Проверяет, доступна ли подсистема ACPI или нет
acpid	Обеспечивает интеллектуальное управление питанием и используется для уведомления программ пользовательского пространства о событиях ACPI.
acpi	Отображение состояния батареи и другой информации подсистемы ACPI (сокр. от "Advanced Configuration and Power Interface")
ac	Отображает статистику о времени подключения пользователей (в часах) на основе входов/выходов из Системы
addgroup	Добавляет группу в Систему
addr2line	Конвертирует адреса в имена файлов и номера строк
adduser	Добавляет нового пользователя в Систему
agetty	Linux-версия getty, которая представляет собой Unix-программу, работающую на главном компьютере и управляющую физическими или виртуальными терминалами для обеспечения многопользовательского доступа
agrep	Ищет в файле записи, содержащие строки, которые точно или приблизительно соответствуют заданному шаблону
alias	Создание или удаление псевдонима для команды или серии команд
alsactl	Управление звуковым драйвером ALSA
amixer	Консольный микшер звуковой карты, работающей под управлением ALSA (сокр. от "Advanced Linux Sound Architecture" – набор драйверов и утилит для поддержки звука в ОС)
amp	Проигрывает заданные MPEG аудио файлы или (если вызывается с опцией -convert) распаковывает файл в wave-формат
aplay	Консольный аудиоплеер для звуковых карт, работающих под управлением ALSA
apm	Запрос к подсистеме расширенного управления питанием (Advanced Power Management APM) BIOS

Команда	Описание
apropos	Помогает пользователю найти команду по ключевому слову, связанному с ней. Показывает краткое описание команд, в которых присутствует искомое слово
arch	Отображает информацию об архитектуре компьютера
aria2c	Сверхбыстрая утилита загрузки
arp	Управление системным ARP-кешем (сокр. от "Address Resolution Protocol"). Также позволяет создать его полный дамп
ar	Используется для создания, изменения и извлечения файлов из архивов
aspell	Используется в качестве средства проверки орфографии
as	Ассемблер проекта GNU; используется компилятором GCC
atd	Демон планировщика заданий, запускающий задания, поставленные в очередь с помощью команды at
atq	Отображает список отложенных заданий, запланированных пользователем
atrm	Удаление указанных заданий, добавленных через команду at. Чтобы удалить выбранное задание, необходимо передать команде его номер
attr	Расширенные атрибуты объектов xfs
at	Выполняет команды в заданное время
autoconf	Создание конфигурационных скриптов для пакетов с исходным кодом
autoheader	Создание файла шаблона операторов #define или любого другого заголовка шаблона
automake	Автоматическое создание makefile-ов (файлов, соответствующих стандартам кодирования GNU)
autoreconf	Создание автоматически собираемого исходного кода
autoupdate	Обновление файла configure.in до более новой версии Autoconf
awk	Язык поиска и обработки шаблонов
B	
badblocks	Поиск на диске сбойных блоков
banner	Вывод большими буквами строки ASCII-символов в стандартный вывод
base64	Base64 – кодирование/декодирование данных и стандартный вывод
basename	Удаляет информацию о каталоге и суффиксы из имен файлов, т.е. Выводит имя файла с удалением всех компонентов каталога
bash	Запуск командной оболочки bash (сокр. от "bourne again shell"). Данная оболочка является стандартным интерпретатором в большинстве Linux-дистрибутивов
batch	Используется для считывания команд из стандартного ввода или указанного файла и их выполнения при разрешенном уровне нагрузки Системы, т.е. когда средняя нагрузка падает ниже 1.5.
bc	Консольный калькулятор
bdiff	Как и команда diff, применяется для поиска отличий, но в больших файлах

Команда	Описание
bg	Используется для восстановления на передний план задачи, выполняемой в фоновом режиме
biff	Система почтовых уведомлений, которая оповещает пользователя в командной строке о появлении новых писем
bind	Встроенная команда оболочки bash, которая используется для установки привязок клавиш и переменных Readline
bison	Генератор парсера, схожий с yacc
break	Используется для завершения выполнения циклов for, while и until
builtin	Выполняет встроенную команду оболочки. Используется при замене встроенной команды оболочки некоторой функцией, но при этом нужна функциональность встроенной команды внутри самой функции
bye	Аналог команды exit, используется для завершения сессии или выхода из терминала
bzcat	Декомпрессия файлов в stdout
bzcmp	Вызов утилиты cmp для сжатых файлов формата .bzip
bzdiff	Сравнение сжатых файлов формата .bzip2
bzgrep	Поиск шаблона или выражения, но внутри сжатого файла формата .bzip2
bzip2, bunzip2	Сжатие и распаковка файлов
bzip2recover	Восстановление поврежденных bzip2-файлов
bzless	Похожая на команду bzmore, но имеет гораздо больше функций. Bzless не нужно читать весь входной файл перед запуском, поэтому с большим файлом она запускается быстрее, чем текстовые редакторы (например, vi)
bzmore	Это фильтр, который позволяет поэкранно просматривать как сжатые (.bzip2), так и простые текстовые файлы
C	
calendar	Служба напоминаний
cal	Просмотр календаря определенного месяца или целого года. По умолчанию показывает календарь текущего месяца
cancel	Остановка вывода информации о работе задачи
case	Альтернатива нескольким операторам if/elif (когда используется одна переменная)
cat	Считывает данные из файла и выдает их содержимое в качестве выходных данных. Также способен объединять данные из нескольких файлов
ccrypt	Инструмент командной строки для шифрования и дешифрования данных
cc	Используется для компиляции кода языка C и создания исполняемых файлов.
cd	Изменение текущего рабочего каталога
cdrecord	Запись компакт дисков (в том числе и музыкальных) из образа диска

Команда	Описание
`cfdisk`	Утилита для просмотра и управления таблицей разделов диска
`chage`	Утилита для просмотра и изменения информации об истечении срока действия пароля пользователя
`chattr`	Изменение атрибутов файла в каталоге
`chcon`	Изменение контекста безопасности файла
`chdir`	Смена рабочей директории (аналог команды `cd`)
`checkeq`	Процессор языка программирования для описания уравнений и проведения сравнений
`checknr`	Проверка nroff- и troff-файлов на ошибки
`chfn`	Позволяет легко изменить имя пользователя и другие детали
`chgrp`	Изменение группы, владеющей файлом или каталогом
`chkconfig`	Перечисление текущей информации о запуске служб или какой-либо конкретной службы, а также обновления настроек уровня запуска службы
`chmod`	Используется для управления разрешениями на заданный файл/каталог
`chown`	Изменение владельца файла или группы
`chpasswd`	Смена пароля одновременно для нескольких пользователей
`chroot`	Изменение корневого каталога
`chrt`	Управление атрибутами реального времени процесса
`chsh`	Изменение оболочки входа пользователя (текущего shell)
`chvt`	Переключение между различными доступными терминалами TTY (сокр. от "Teletypewriter")
`cksum`	Отображение и вычисление значения контрольной суммы файла или CRC (сокр. от "Cyclic Redundancy Check"), его размера в байтах и имени в стандартном выводе
`clear`	Очистка экрана терминала
`cmp`	Побайтовое сравнение двух файлов. Помогает выяснить, идентичны ли два сравниваемых файла или нет
`colcrt`	Форматирование вывода текстового процессора таким образом, чтобы его можно было просматривать на дисплеях с ЭЛТ (сокр. от "Электронно-Лучевая Трубка")
`colrm`	Удаляет выбранные столбцы из строк файла. Столбец определяется как один символ в строке. Входные данные считывается со стандартного входа. Результат записывается в стандартный вывод
`column`	Форматирование отображения содержимого файла в виде столбцов
`col`	Фильтрует переводы строк из входного потока
`comm`	Сравнивает два отсортированных файла построчно и записывает в стандартный вывод: строки, которые являются общими, и строки, которые являются уникальными
`compress`	Уменьшение размера файла. После сжатия файл приобретает расширение .z.

Команда	Описание
continue	Пропуск текущей итерации в циклах for, while и until
cpio	Обработка архивных файлов, таких как *.cpio или *.tar. Данная команда может копировать файлы в архивы и из архивов
cpp	Автоматически используется компилятором C для преобразования программы перед компиляцией
cp	Копирование файлов и каталогов
crontab	Список команд, которые требуется выполнять по регулярному расписанию, а также имя команды, используемой для управления этим списком
cron	Утилита, автоматизирующая выполнение запланированной задачи в заданное время
csh	Командный интерпретатор C Shell
csplit	Разделение любого файла на множество частей
ctags	Позволяет быстро получить доступ к файлам (например, быстро увидеть определение функции)
cupsd	Планировщик подсистемы вывода CUPS
curl	Утилита для передачи данных на сервер или с сервера с использованием любого из поддерживаемых протоколов
cut	Утилита для вывода заданной части строк каждого заданного файла
cu	Отправка сигнала через терминал другой Системе
cvs	Хранение истории изменений файла. При повреждении файла или других проблемах команда помогает вернуться к предыдущей версии и восстановить файл
D	
date	Отображение и установка системной даты и времени
dc	Вычисление арифметических выражений
dd	Утилита командной строки, основной целью которой является конвертирование и копирование файлов
declare	Объявление переменных и функций, установка атрибутов и отображение их значений
delgroup	Удаление группы из Системы
deluser	Удаление пользователя из Системы
depmod	Формирование списка зависимостей модулей ядра и генерация соответствующих *.map-файлов
deroff	Удаляет из файлов nroff- и troff-конструкции
df	Отображение информации о доступном и использованном дисковом пространстве
dhclient	Утилита для работы с DHCP-протоколом (получение динамического IP-адреса, настройка сетевых интерфейсов и пр.)
diff3	Построчное сравнения трех файлов
diff	Отображение различий в файлах путем их построчного сравнения

Команда	Описание
`dig`	Отображение информации о DNS
`dircmp`	Сравнение содержимого двух каталогов
`dircolors`	Настройка цветов для `ls`
`dirname`	Удаление замыкающих слешей (/) из пути к каталогу и вывод оставшейся части
`dirs`	Отображение списка сохраненных каталогов
`dir`	Перечисление содержимого каталога
`disable`	Деактивирует принтеры, отключая их от запросов на печать, отправляемых командой `lp`
`dnsdomainname`	Вывести или установить системное имя хоста
`dnsquery`	Послать запрос на сервер доменных имен через резолвер
`dmesg`	Выводит сообщения ядра во время начальной загрузки ОС или настраивает их буфер
`dmidecode`	Позволяет получить информацию об аппаратных компонентах Системы, а также другую полезную информацию: характеристики процессора, оперативной памяти (DIMM), детали BIOS и т.д.
`domainname`	Позволяет вывести или установить NIS/YP-доменное имя
`dosfsck`	Диагностирует ФС MS-DOS на наличие проблем и пытается их устранить
`dpost`	Конвертирование файлов из формата troff в postscript
`dstat`	Обычно используется системными администраторами для получения информации о сетевых соединениях, устройствах ввода-вывода, процессоре и т.д.
`dumpe2fs`	Дамп информации ФС ext2/ext3
`dumpkeys`	Выводит информацию о текущей раскладке клавиатуры
`dump`	Резервное копирование ФС на какое-либо запоминающее устройство
`du`	Отслеживание файлов и каталогов, которые занимают чрезмерное количество места на жестком диске
`djview`	Полноценная программа для просмотра файлов djvu
E	
`e2fsck`	Проверка ФС ext2/ext3/ext4
`echo`	Отображение текста/строки на экране, которые передаются в качестве аргумента
`edit`	Текстовый редактор (разновидность редактора для простых пользователей)
`ed`	Запуск построчного текстового редактора с минималистичным интерфейсом
`egrep`	Обрабатывает шаблон как расширенное регулярное выражение и выводит строки, соответствующие шаблону

Команда	Описание
`eject`	Позволяет извлекать съемный носитель (обычно CD-ROM, дискету, ленту, JAZ- или ZIP-диск) с помощью ПО
`elm`	Интерактивная почтовая система
`emacs`	Редактор с простым пользовательским интерфейсом, в котором нет режима вставки. У него есть только режим редактирования
`enable`	Включение/выключение lp-принтеров
`env`	Вывод информации о переменных окружения. Также используется для запуска утилиты или команды в пользовательском окружении
`eqn`	Используется для описания сравнений
`ethtool`	Отображает или позволяет изменить настройки сетевой карты
`eval`	Команда воспринимает переданные ей аргументы как директивы оболочки
`evim`	Простой Vim, позволяет редактировать файл с помощью Vim в незаданном режиме
`exec`	Используется для выполнения команды из самого bash
`exiftool`	Чтение и запись метаданных файлов
`exit`	Закрывает командный интерпретатор с состоянием N. Если N не указано, то состоянием выхода будет состояние последней выполненной команды
`expand`	Позволяет конвертировать табуляции в пробелы в файле, а когда файл не указан, то данные считываются со стандартного ввода
`expect`	Команда, работающая со скриптами, которые ожидают ввода данных от пользователя. Автоматизирует задачу, предоставляя входные данные
`expiry`	Проверяет и изменяет пароль согласно политике устаревания
`export`	Помечает переменные окружения, которые экспортируются в дочерние процессы
`expr`	Вычисляет заданное выражение и отображает результат
`ex`	Текстовый редактор в ОС, который также называется линейным режимом редактора vi
F	
`false`	Ничего не делает; сообщает о безуспешном выполнении
`factor`	Вывод простых множителей заданных чисел (задаваемых как через командную строку, так и через стандартный ввод)
`fc-cache`	Сканирует каталоги шрифтов (и создает их кеш), которые используют fontconfig для обработки шрифтов
`fc-list`	Используется для перечисления доступных шрифтов и стилей шрифтов. Список всех шрифтов можно отфильтровать и отсортировать, применив соответствующую опцию форматирования
`fc`	Используется для перечисления, редактирования или повторного выполнения команд, ранее введенных в интерактивную оболочку
`fdisk`	Диалоговая команда в ОС, используемая для создания и управления таблицей разделов диска
`fetchmail`	Прием почты с серверов POP, IMAP, ETRN, или ODMR

Команда	Описание
ffmpeg	Видеоконвертер
fgrep	Поиск строк в файле
fg	Перемещение фоновой задачи на передний план
file	Вывод типа файла
findsmb	Список всех машин, доступных по SMB-протоколу (сокр. от "Server Message Block")
find	Поиск файлов и каталогов
finger	Подробная информация обо всех пользователях, вошедших в Систему
fmt	Утилита простого форматирования текста
fold	Оборачивает каждую строку во входной файл, чтобы соответствовать заданной ширине, и выводит её на стандартный вывод
foreach	Выполняет набор команд для каждого из элементов заданного массива
for	Используется для многократного выполнения набора команд для каждого элемента, присутствующего в списке
free	Отображение объема свободной и используемой памяти в Системе
fsck.fat	Проверка и восстановление ФС MS-DOS
fsck	Проверка и восстановление ФС Linux
ftp	Интерактивная утилита для доступа к FTP (сокр. от "File Transfer Protocol")
function	Создание функций или методов
fun	Рисование в терминале узоров различного типа
fusermount	Монтирование и демонтирование в ФС FUSE
fuser	Определение процессов, использующих файлы или сокеты
G	
gawk	GNU-версия awk
gcc	Используется для компиляции программ, написанных на языках C, C++, Objective-C и Objective-C++
gdb	Мощный отладчик для программ, написанных на C, C++, Ada, Fortran и пр.
ginstall	Копирование файлов и установка атрибутов
gendiff	Утилита, помогающая создать diff-файл без ошибок
getent	Получение элементов из базы данных
getfacl	Получение списков контроля доступа к файлам
getfattr	Получить расширенные атрибуты объектов ФС
git	Система управления версиями
glxinfo	Показать информацию о реализации GLX

Команда	Описание
gpasswd	Администрирование файлов /etc/group и /etc/shadow
gpg	Инструмент для шифрования и цифровой подписи
gpgv	Программа проверки подписей
gprof	Отображение данных по профилированию программы
grep	Поиск в файле определенного шаблона символов и вывод всех строк, содержащих этот шаблон
grodvi	Преобразует вывод groff в форматы tex и dvi
groff	Frontend-приложение для системы форматирования документов groff
groupadd	Создание новой группы
groupdel	Удаление существующей группы
groupmod	Модификация или смена существующей группы
groups	Отображение списка групп, в которых состоит пользователь
grpck	Проверка целостности информации о группах, а именно: все записи в /etc/group и /etc/gshadow имеют правильный формат и содержат допустимые данные
grpconv	Преобразует пароли пользователей и групп в/из защищенную форму
gs	Команда вызывает Ghostscript-интерпретатор языка Adobe Systems postscript и формата PDF (сокр. от "Portable Document Format")
gunzip	Сжатие или распаковка файлов
gview	Версия редактора vi с графическим интерфейсом. Запускается в новом окне
gvim	Синоним для команды gview
gzexe	Сжатие исполняемых файлов, а также их автоматическое распаковывание в момент выполнения
Gzip	Сжатие файлов
g++	Используется для предварительной обработки, компиляции, сборки и линкинга исходного кода при создании исполняемого файла
H	
halt	Указание аппаратному обеспечению компьютера остановить все исполняющиеся в данный момент процессы. Основное применение перезагрузка или выключение Системы
hash	Доступ к хеш-таблице недавно выполненных программ
hdparm	Получение информации о жестком диске, изменении интервалов записи, настроек DMA (сокр. от "Direct Memory Access")
head	Выводит N первых строк файла
help	Отображает информацию о встроенных командах оболочки
hexdump	Фильтр, который отображает указанные файлы или стандартный ввод, если файлы не указаны, в заданном пользователем формате

Команда	Описание
history	Отображает историю команд, которые были введены с момента начала сессии
hostid	Отображает числовой идентификатор текущего хоста.
hostnamectl	Может использоваться для запроса и изменения имени хоста Системы и связанных с ним параметров
hostname	Выдает или устанавливает имя компьютера
host	Утилита для работы с DNS-запросами
htop	Консольная утилита, которая позволяет пользователю интерактивно (в режиме реального времени) отслеживать список запущенных процессов
hwclock	Запрос и установка аппаратных часов (сокр. "rtc" от "real-time clock")
I	
iconv	Преобразование некоторого текста из одной кодировки в другую
id	Выводит информацию о текущем пользователе и группе
ifconfig	Конфигурирование сетевого интерфейса
ifdown	Отключает сетевой интерфейс
ifquery	Позволяет выбрать информацию о сетевом интерфейсе
iftop	Инструмент анализа сети, используемый системными администраторами для просмотра статистики, связанной с пропускной способностью канала передачи данных
ifup	Задействует (поднимает) сетевой интерфейс, позволяя ему передавать и получать данные
imagemagick	Набор программ для чтения и редактирования файлов множества графических форматов
import	Создание скриншота экрана (всего экрана или только его части) с последующим сохранением в файл
info	Позволяет читать документацию в формате info
insmod	Программа для активации модулей ядра
install	Копирует файлы и устанавливает атрибуты
inxi	Показывает информацию о Системе
iostat	Мониторинг статистики I/O-операций для устройств и разделов
iotop	Отображение статистики работы процессов с дисками
ipcrm	Удаление IPC-ресурсов (сокр. от "Inter-Process Communication") и связанных с ними данных
ipcs	Вывод информации об IPC-ресурсах
iptables-save	Сохраняет текущие правила iptables в указанном файле
iptables	Настройка правил брандмауэра, входящего в состав ядра ОС

Команда	Описание
`ip`	Отображение и управление сетевыми подключениями, маршрутизацией и пр.
`iwconfig`	Отображение параметров и статистики беспроводной связи, которые извлекаются из /proc/net/wireless
`iwlist`	Используется для отображения дополнительной информации от беспроводного сетевого интерфейса, не показываемой iwconfig
J	
`jobs`	Отображение состояния заданий в текущей сессии
`join`	Утилита для соединения строк двух файлов на основе ключевого поля, присутствующего в обоих файлах
`journalctl`	Используется для просмотра логов, собранных systemd. Systemd собирает логи в бинарном формате
K	
`kernelversion`	Выводит основную версию ядра
`killall`	Посылает сигнал kill всем активным процессам
`kill`	Посылает процессу сигнал завершения работы
`ksh`	Командный интерпретатор (оболочка) Korn Shell
L	
`last`	Отображение списка всех пользователей, вошедших и вышедших из Системы с момента создания файла /var/log/wtmp
`lastb`	Отображение списка последних вошедших в Систему пользователей
`ldd`	Отображение зависимостей разделяемых библиотек
`ld`	Редактор ссылок на библиотеки для объектов
`less`	Поэкранное чтение содержимого текстового файла
`let`	Вычисление арифметических выражений для переменных оболочки
`lexgrog`	Анализирует заголовочную информацию справочных страниц
`link`	Создание жесткой ссылки на файл
`ln`	Создание символьных ссылок между файлами
`locate`	Поиск файлов по имени
`login`	Вход в Систему
`logname`	Отображает логин пользователя
`logout`	Аналог `lo`
`look`	Выводит строки, начинающиеся с заданной подстроки
`losetup`	Настройка и управление виртуальными loop-устройствами
`lo`	Завершает работу с командной оболочкой

Команда	Описание
`locale`	Выдаёт информацию о локали
`localedef`	Компилирует файлы определения локали
`lpq`	Просмотр очереди печати
`lpr`	Локальная печать
`lprm`	Команда удаления заданий из очереди
`lsblk`	Отображение сведений о блочных устройствах
`lsb_release -a`	Отображение информации о дистрибутиве
`lshw`	Генерация подробной информации об аппаратной конфигурации Системы на основе файлов из каталога /proc
`lsmod`	Отображение модулей ядра ОС, которые в данный момент загружены
`lsof`	Отображает информацию о том, какие файлы используются тем или иным процессом
`lspci`	Отображает список всех устройств на шине PCI
`lsusb`	Отображение информации о USB-шинах и устройствах, подключенных к ним
`ls`	Отображение списка содержимого каталога
`lsattr`	Выводит атрибуты файла второй расширенной ФС Linux
`lynx`	Браузер
`lzcat`	Посмотреть содержимое файла, сжатого LZMA (сокр. от "Lempel-Ziv-Markov chain-Algorithm")
`Lzma`	Сжать или распаковать файл по алгоритму LZMA
`lzop`	Сжимает или разжимает файлы
M	
`mach`	Отображение информации о типе процессора
`mailq`	Сводка почтовых сообщений, поставленных в очередь для дальнейшей доставки
`mailx`	Интерактивная система обработки сообщений электронной почты
`make-ssh-known-hosts`	Создание файла ssh_known_hosts из данных DNS
`make`	Автоматическое определение частей большой программы, которые должны быть перекомпилированы, и выдача команды для их перекомпиляции
`manconv`	Изменяет кодировку справочной страницы
`manpath`	Определяет путь поиска справочных страниц
`man-recode`	Изменяет кодировку справочных страниц
`man`	Отображение руководства пользователя по любой команде, которую можно запустить в терминале

Команда	Описание
mc	Файловый менеджер
md5sum	Проверка целостности данных с помощью алгоритма хеширования MD5
memusagestat	Генерирует график по данным профилирования памяти
memusage	Исследует использование памяти программы
merge	Слияние содержимого трех файлов
mesg	Отправка сообщений в другой терминал
mkdir	Создание одного или сразу нескольких каталогов
mkfs	Создание ФС в выбранном разделе
mknod	Создает специальные блочные или символьные файлы
mkswap	Создание файла (или раздела) подкачки (swap)
mkvextract	Извлечение дорожки из файлов Matroska(TM) в другие файлы
mkvinfo	Запись информации об элементах в файлы Matroska(TM)
mkvmerge	Упаковка потоков мультимедиа в файл Matroska(TM)
mkvpropedit	Изменяет свойства существующих файлов Matroska(TM) без сложного ремультиплексирования
mkvtoolnix-gui	Графический интерфейс для mkvmerge(1), включающий редакторы глав и заголовков
modeprobe	Программа для загрузки и выгрузки модулей из ядра ОС
modinfo	Отображение информации о модуле ядра ОС
more	Поэкранный просмотр текстовых файлов в командной строке
mount	Примонтировать ФС в директорию
mpstat	Создание статистического отчета о работе процессора (или по каждому процессору, если их несколько)
mtrace	Трактует журнал трассировки malloc
mt	Управление работой накопителя с магнитной лентой
mv	Перемещение внутри файловой системы одного или нескольких файлов, или каталогов из одного места в другое
mysqldump	Утилита для создания резервной копии базы данных mysql
mysql	Программа для управления базой данных mysql
N	
nc	Мощная утилита, которая используется для решения различных задач, связанных с сетевыми TCP- и UDP-подключениями
nethogs	Группирует пропускную способность для каждого процесса

Команда	Описание
netstat	Отображает различную информацию, связанную с сетью (например, сетевые соединения, таблицы маршрутизации, статистика интерфейса и пр.)
networkctl	Запрос состояния сетевых ссылок
newgrp	Предоставляет пользователю права новой группы на время
newsbeuter	Программа чтения RSS/Atom для текстовых терминалов
nice	Запуск программы с измененным приоритетом
niscat	Отображение таблиц и объектов NIS+
nischmod	Изменение прав доступа к объекту NIS+
nischown	Изменение владельца объекта NIS+
nischttl	Изменение значения времени жизни объекта NIS+
nisdefaults	Отображение заданных по умолчанию значений NIS+
nistbladm	Команда администрирования таблицы NIS+
nisdomainname	Вывести или установить системное имя хоста
nl	Печатает каждый ФАЙЛ на стандартный вывод, добавляя номера строк
nmap	Инструмент исследования сети и сканер безопасности/портов
nmcli	Управление networkmanager. Также может применяться для отображения состояния сетевого устройства, создания, редактирования, активации/деактивации и удаления сетевых подключений
nohup	Продолжить выполнение команды, когда сессия терминала будет завершена
nroff	Приложение для форматирования документов
nslookup	Инструмент сетевого администрирования, связанного с DNS
O	
od	Преобразование содержимого входных данных в восьмеричный и другие форматы
onintr	Отображение информации об аппаратных прерываниях
on	Выполнение команды на удаленной системе, но с локальным окружением
osec	Выполняет контроль целостности Системы
P	
pack	Сжатие файлов по алгоритму Хаффмана
pact	Вывод содержимого сжатого текстового файла
pagesize	Отображение размера страницы в памяти
parted	Программа для разметки диска
partprobe	Информирование ОС об изменениях в таблице разделов

Команда	Описание
passwd	Изменение паролей учетных записей пользователей
paste	Соединение файлов (параллельным слиянием) путем вывода результирующих строк, состоящих из строк каждого указанного файла, разделенных табуляцией в качестве разделителя, в стандартный вывод
pathchk	Проверить правильность и портируемость имени файла
pax	Чтение и запись файловых архивов и копирование иерархий каталогов
pdftk	Простой редактор pdf-файлов
perl	Интерпретатор Perl-скриптов
pgrep	Позволяет находить идентификаторы процессов работающей программы на основе заданных критериев
pg	Фильтр для постраничного просмотра содержимого текстовых файлов
pico	Простой текстовый редактор в стиле Pine Composer
pidof	Определение идентификаторов процессов конкретной запущенной программы
pine	Программа для Интернет-новостей и электронной почты
ping	Проверка сетевого подключения между хостом и сервером/другим хостом
pinky	Команда поиска информации о пользователе, которая дает подробную информацию обо всех пользователях, вошедших в Систему. В отличие от finger можно обрезать интересующую вас информацию
pkexec	Выполнить команду от имени другого пользователя
pkill	Завершить процесс по его имени
pmap	Отображение информации об адресном пространстве процесса
postconf	Конфигурационный инструментарий Postfix
postdrop	Агент отправки почты Postfix
postfix	Программа управления почтовой системой Postfix
postkick	Подстегнуть службу Postfix
postlock	Блокирует почтовую папку и выполняет команду
postlog	Postfix-совместимый инструментарий журналирования
postsuper	Суперинтендант Postfix
poweroff	Выключение Системы
preparetips5	Извлечение текста из файлов с полезными советами
printenv	Отображение переменных окружения
printf	Отображение отформатированных данных (числа, строки и пр.)
procmail	Автономный обработчик почты

Команда	Описание
prtstat	Выводит статистику по процессу
pr	Преобразование текстовых файлов для вывода
pstree	Отображение дерева процессов
ps	Вывод информации об активных процессах
ptx	Составляет оглавление файла
pvs	Производит форматированный вывод информации о физических томах
pwck	Проверяет целостность файлов паролей
pwd	Отображает полный путь к текущему рабочему каталогу
Q	
qrencode	Создает QR-коды и сохраняет в виде картинки на диск
quit	Завершение сеанса командной оболочки
R	
ranlib	Генерирует индекс к содержимому архива и сохраняет его в архиве
rb	Получение файлов по протоколам XMODEM, YMODEM, ZMODEM
rcp	Копирование файлов с одного компьютера на другой
readelf	Получение информации о файлах формата ELF (сокр. от "Executable and Linkable Format")
readlink	Отображение значения символьной ссылки
read	Считывание строки со стандартного ввода
reboot	Перезагрузка Системы
redshift	Изменяет цветовой профиль экрана в зависимости от времени суток
red	Запуск редактора ed в режиме прокрутки текста
rename	Переименование файлов
repeat	Повторять выполнение команды нужное количество раз
replace	Утилита замены строк в файлах
reset	Инициализация терминала. Полезно в тех ситуациях, когда завершившаяся программа оставила терминал в "ненормальном" состоянии
restore	Восстановление файлов из резервной копии, созданной с помощью dump
rev	Реверсивное изменение строк файла
rlogin	Удаленный вход в Систему
rmdir	Удаление пустых каталогов

Команда	Описание
`rmmod`	Удаление модуля из ядра
`rm`	Удаление файлов, каталогов, символьных ссылок и пр.
`route`	Используется для работы с IP-адресами и таблицей маршрутизации
`rpcgen`	Компилятор протокола RPC
`rpcinfo`	Отображает информацию об RPC (сокр. от "Remote Procedure Call")
`rpm`	Менеджер пакетов RPM
`rsh`	Удаленный командный интерпретатор
`rsync`	Позволяет синхронизировать файлы и каталоги между двумя местоположениями. Ведет себя почти так же, как rcp, но имеет гораздо больше опций и использует протокол удаленного обновления для значительного ускорения передачи файлов при обновлении целевого файла
`runcon`	Запуск команды с указанным контекстом безопасности
`rview`	Улучшенный Vi, текстовый редактор для программистов
`rvim`	Улучшенный Vi, текстовый редактор для программистов
`rx`	Получение файлов по протоколам XMODEM, YMODEM, ZMODEM
`rz`	Получение файлов по протоколам XMODEM, YMODEM, ZMODEM
S	
`s2p`	Конвертер sed-скриптов в Perl
`sar`	Мониторинг ресурсов ОС, таких как загрузка процессора, использование памяти, нагрузка на устройства ввода-вывода и т.д.
`scp`	Безопасное копирование файлов между серверами
`screen`	Обеспечивает возможность запуска и использования нескольких сеансов оболочки из одной SSH-сессии
`scriptreplay`	Воспроизведение активности терминала, сохраненной с помощью команды `script`
`script`	Используется для записи всего, что происходит в терминале
`sdiff`	Находит различия между двумя файлами и объединяет их в интерактивном режиме
`sed`	Потоковый редактор для фильтрации и преобразования текста
`select, pselect`	Создание нумерованного меню, из которого пользователь может выбрать нужный параметр
`sendmail`	Отправка электронного письма
`sensors`	Мониторинг температуры и напряжения
`seq`	Генерирует последовательность чисел с заданным шагом
`service`	Утилита для запуска и управления службами

Команда	Описание
sestatus	Проверка того, включен ли selinux (модуль безопасности ОС)
setfacl	Настройка списков контроля доступа к файлам
setsid	Запуск программы в новом сеансе
set	Показывает весь набор переменных в окружении
sfdisk	Программа для разметки дисков
sftp	Клиент для передачи файлов по протоколу SFTP
shift	Сдвигает/перемещает аргументы командной строки на одну позицию влево
showkey	Отображает код каждой кнопки, нажимаемой на клавиатуре
shred	Переписывает несколько раз указанные файлы для того, чтобы сделать более трудоёмким процесс восстановления данных даже в случае использования специального оборудования для восстановления
shuf	Перемешивание строк в файле
shutdown	Безопасное выключение Системы
sh	Командная оболочка Bourne Shell
sleep	Задержка на заданное количество времени
slogin	Псевдоним SSH-клиента, применяемый для удаленного вход в Систему через SSH
smartctl	Утилита управления и мониторинга для SMART-дисков
smbclient	Ftp-подобный клиент для доступа к ресурсам SMB/CIFS
sort	Сортировка строк в текстовых файлах
source	Чтение и выполнение содержимого файла (обычно набора команд), передаваемого в качестве аргумента в текущем скрипте оболочки
spell	Проверка орфографии
split	Разделение больших файлов на более мелкие
sha1sum	Вычисляет и сверяет криптографические контрольные суммы по алгоритму SHA1 (устарел)
sshfs	Позволяет смонтировать файловую систему удалённого сервера с помощью протокола SSH
ssh	Протокол, используемый для безопасного подключения к удаленному серверу/Системе
ssh-add	Добавляет RSA- или DSA-идентификацию в аутентификационного агента
ssh-agent	Агент аутентификации
ssh-copy-id	Инсталлирует ваши файлы публичной идентификации на удаленной машине
ssh-keygen	Создание, обслуживание и преобразование аутентификационных ключей
ssh-keyscan	Собирает публичные ключи SSH

Команда	Описание
ss	Просмотр информации о сетевых подключениях
startx	Инициализация сессии X-сервера
stat	Отображение статистики файла или ФС
stop	Остановка фоновой задачи
strace	Один из самых мощных инструментов мониторинга и диагностики, позволяющий проводить трассировку системных вызовов и сигналов
strip	Позволяет удалить отладочную информацию из исполняемых файлов
stty	Отображает и устанавливает параметры терминала
sudo	Используется в качестве префикса некоторой команды, которую разрешено запускать только суперпользователю
sum	Вычисление контрольной суммы и подсчета блоков в файле
su	Авторизация от имени другого пользователя
swapoff	Отключает файл (или раздел) подкачки
swapon	Задействует файл (или раздел) подкачки
sync	Синхронизация данных на диске с данными в памят
systemctl	Используется для проверки и контроля состояния systemd и управления службами.
sx	Передача файлов по протоколам XMODEM, YMODEM, ZMODEM
sz	Передача файлов по протоколам XMODEM, YMODEM, ZMODEM
T	
tabs	Останавливает работу вкладок в терминале
tac	Объединение и вывод файлов в обратном порядке
tail	Выводит N последних строк файлов
talk	Программа визуальной коммуникации, которая копирует строки с терминала в терминал другого пользователя
tar	Используется для создания архива и извлечения архивированных файлов
tcopy	Копирование магнитной ленты
tcpdump	Консольный анализатор сетевого траффика
tcsh	Командный интерпретатор TENEX C Shell
tee	Считывает данные из стандартного устройства ввода и записывает их на стандартное устройство вывода или в файл
telnet	Используется для связи с другим хостом по протоколу TELNET (сокр. от "Teletype Network")
terminfo	База данных о возможностях терминала
test	Проверяет типы файло

Команда	Описание
timex	Время выполнения команды (с отображением более подробной информации).
time	Выполняет команду и по её завершению отображает статистику о затраченных временных ресурсах на её исполнение
todos	Конвертирование текстовых файлов Unix в формат DOS
top	Отображение и обновление информации о топе процессов CPU
touch	Используется для создания, изменения и модификации временных меток файла
tput, reset	Инициализирует терминал или запрашивает базу данных terminfo
tracepath	Отображает маршрут по сетевым узлам с MTU (сокр. от "Maximum Transmission Unit")
traceroute	Отображает маршрут, по которому пакет достигает хоста
tree	Список содержимого каталогов в древовидном формате
truncate	Удаляет все содержимое файла. Она не удаляет файл, а оставляет его как файл с нулевым байтом на диске. Если воспользоваться командой усечения, права доступа к файлу и право собственности на него сохранятся
tr	Преобразование или удаление символов.
true	Ничего не делает, всегда завершается успешно
tsort	Выполняет топологическую сортировку
tty	Отображает информацию, связанную с терминалом (имя файла терминала, подключенного к стандартному входу)
type	Используется для описания того, как будет интерпретироваться каждый аргумент, если он будет использоваться в качестве имени команды
U	
umask	Устанавливает маску прав для режима создания файла
umount	Позволяет размонтировать Ф
unalias	Удаление псевдоним
uname	Отображает информацию о Системе
uncompress	Распаковка сжатого файла
unexpand	Преобразует каждый пробел в табуляцию, записывая результат в стандартный вывод
uniq	Отображает или пропускает повторяющиеся строки
unix2dos	Преобразует текстовый файл Unix в формат DOS
unlink	Удаление ссылки на файл
unlzma	Извлечение файлов из архива, сжатого командой Lzma
unpack	Извлечение файлов из архива, сжатого командой pack
unxz	Распаковка .xz-архива

Команда	Описание
unzip	Распаковка .zip-архива
uptime	Выводит информацию о времени работы Системы
useradd	Добавление учетных записей пользователей в Систему.
userdel	Удаление учетной записи пользователя и связанных с ней файлов
usermod	Изменяет учётную запись пользователя
username	Получение имени пользователя
users	Отображение имен пользователей, вошедших в Систему на данный момент
V	
vacation	Автоответчик для электронной почты
vdir	Отображение списка содержимого каталога
vi	Текстовый редактор
vim	Свободный текстовый редактор, созданный на основе более старого vi
vimdiff	Позволяет редактировать две или три версии файла с помощью Vim с отображением различий
vimtutor	Учебник по Vim
vmstat	Команда мониторинга производительности Системы, предоставляющая информацию о процессах, памяти, файле подкачке, активности процессора и т.д.
vlock	Программа для блокирования виртуальных консолей
vnstat	Используется системными администраторами для мониторинга параметров сети: загруженность каналов, входящий/исходящий трафик и пр.
W	
wall	Отображает в терминале сообщение для всех пользователей, вошедших в Систему
watch	Используется для запуска и слежки за программой, отображая её вывод в терминал
wc	Определение количества строк, слов, байтов и символов в файлах
wget	Утилита для загрузки файлов из Интернета
whatis	Получение однострочного описания искомой команды
whereis	Отображает полный путь к исполняемому файлу программы (или к исходным файлам, если они присутствуют в Системе)
which	Поиск исполняемого файла, связанного с заданной командой, путем нахождения его в переменной окружения PATH
whoami	Отображает имя текущего пользователя.
whois	Отображение доступной информации об Интернет-ресурсе (например, о сайте)
who	Сообщает обо всех зарегистрировавшихся в Системе пользователях

Команда	Описание
wipe	Безопасное стирание информации с магнитных носителей
write	Позволяет пользователю коммуницировать с другими пользователями, копируя строки с одного терминала в другие
w	Отображает пользователей, вошедших в Систему и их процессы
X	
xargs	Преобразует входные данные, полученные от стандартного ввода, в аргументы команды
xdg-open	Используется для открытия файла или URL-адреса в предпочтительном приложении пользователя
xfd	Отображение всех символов шрифта X-сервера
xinit	Программа для инициализации системы X Windows
xhost	Настройка прав доступа к X-серверу
xkill	Отключить клиента от его X-ресурса
xlsfonts	Отображение всех шрифтов X-сервера
xmlstarlet	Командная строка инструментария XML/XSLT
xmodmap	Утилита для изменения раскладки клавиш клавиатуры и мыши в Xorg
xrandr	Настройка режимов работы с дисплеями (мониторами)
xrdb	Управление базой данных ресурсов X-сервера
xset	Изменение значения переменной X-сервера
xzcat	Просмотр содержимого текстового файла, сжатого командой xz
xz	Сжатие файла в .xz-формат
xxd	Создаёт представление файла в виде шестнадцатеричных кодов или выполняет обратное преобразование
Y	
yacc	Утилита, служащая стандартным генератором синтаксических анализаторов (парсеров) в Unix-системах
yes	Выводит строку несколько раз до тех пор, пока команде не будет послан сигнал kill
ypdomainname	Вывести или установить системное имя хоста
yppasswd	Изменение пароля базы данных NIS
Z	
zcat	Отображение содержимого файла, сжатого командой zip
zdiff	Вызывает diff для файлов, сжатых командой Gzip
zdump	Получение информации о часовом поясе
zforce	Присвоить расширение .gz всем архивам Gzip

Команда	Описание
`zgrep`	Поиск выражений в заданном файле, даже если он сжат
`zipcloack`	Зашифровать .zip-файл
`zipinfo`	Вывести информацию о .zip-файле
`zipnote`	Просмотр и изменение комментариев к .zip-файлам
`zipsplit`	Объединение нескольких .zip-файлов
`zip`	Сжать файлы в архив
`znew`	Перепаковать .Z-файлы в .gz-файлы

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

Сокращение	Определение
БД	База данных
ВМ	Виртуальная машина
ЗПС	Замкнутая программная среда
ИТ	Информационные технологии
ОС	Операционная система
ПК	Персональный компьютер
ПО	Программное обеспечение
СДЗ	Средство доверенной загрузки
УЗ	Учетная запись
СУБД	Система управления базой данных
ЦС	Центр сертификации
ФС	Файловая система
ACL	Access Control List – список контроля доступа
AIDE	Advanced Intrusion Detection Environment – усовершенствованная система обнаружения вторжений
API	Application programming interface – программный интерфейс приложения
BIOS	Basic Input/Output System – базовая подсистема ввода-вывода
CA	Certification authority – центр сертификации (удостоверяющий центр)
CIFS	Common Internet File System – протокол, являющийся частью протокола SMB, для удаленного подключения нескольких платформ – Windows, Linux или macOS, для идентификации и чтения файлов системы NTFS в Windows и HFS в macOS, чтобы совместно использовать эти файлы и принтеры между машинами с Windows и Linux
CSV	Comma separated values – текстовый формат, предназначенный для представления табличных данных
DHCP	Dynamic host configuration protocol – протокол динамической настройки сетевой конфигурации хоста
DNS	Domain name system – система доменных имен
ENC	External node classifier – классификатор внешних хостов
ESR	Extended support release – релиз с расширенной (долговременной) поддержкой
Ext4	Fourth Extended Filesystem – журналируемая файловая система
Ext3	Third Extended Filesystem – журналируемая файловая система
GUI	Graphical User Interface — это графический пользовательский интерфейс,
HTTP	Hypertext transfer protocol – протокол передачи гипертекста

Сокращение	Определение
HTTPS	Hypertext transfer protocol secure – защищенная версия протокола передачи гипертекста
IP	Internet protocol – протокол межсетевого взаимодействия
IPv4	Internet Protocol Version 4 – версия 4 интернет-протокола
IPv6	Internet Protocol Version 6 – версия 6 интернет-протокола
IPP	Internet Printing Protocol – Интернет-протокол печати
IPA	Identity, policy and audit – система идентификации, политик и аудита (СИПА)
IPC	Interprocess Communication – средство межпроцессного взаимодействия
LDAP	Lightweight directory access protocol – протокол доступа к каталогам
LDAPS	Lightweight directory access protocol secure – защищенная версия протокола доступа к каталогам
LUKS	Linux Unified Key Setup – система шифрования дисков, которая хранит данные в зашифрованном физическом разделе
MAC	Media access control – уникальный идентификатор сетевого оборудования
MBR	Master boot record – главная загрузочная запись
MTA	Mail transfer agent – агент передачи почтовых сообщений
NTFS	New Technology File System — это проприетарная файловая система, разработанная компанией Microsoft.
NTP	Network time protocol – протокол сетевого времени
SMB	Server Message Block – протокол соединения с устройствами со старыми версиями ОС Microsoft для подключения к серверам и обмена файлами
SMTP	Simple mail transfer protocol – протокол передачи почтовых сообщений
SSH	Secure shell – защищенная оболочка
SSL	Secure sockets layer – уровень защищенных сокетов
TCP	Transmission control protocol – протокол управления передачей данных
TFTP	Trivial file transfer protocol – протокол передачи файлов
TTY	Teletype – телетайп
UDP	User datagram protocol – протокол пользовательских датаграмм
URL	Uniform resource locator – сетевой адрес ресурса
USB	Universal Serial Bus – стандарт для подключения и передачи данных между ПК и периферийными устройствами
XFS	журналируемая файловая система
YAML	Yet another markup language – язык разметки