

Войтов Н. М.

Администрирование ОС Red Hat Enterprise Linux

Учебный курс



Москва, 2011

УДК 004.451.9Linux 5
ББК 32.973.26-018.2
В65

Войтов Н. М.

В65 Курс RH-133. Администрирование ОС Red Hat Enterprise Linux. Конспект лекций и практические работы ver. 1.10. – М.: ДМК Пресс, 2011. – 192 с.: ил.

ISBN 978-5-94074-677-5

Эта книга позволит читателям получить знания и навыки, необходимые для успешного системного и сетевого администрирования операционной системы Red Hat Enterprise Linux 5 (RHEL), а также для решения задач, связанных с информационной безопасностью.

Издание состоит из теоретической и практической частей, которые вместе позволяют получить систематизированные знания об ОС RHEL и умения решать практические задачи. Теоретическая часть раскрывает принципы работы системы, нюансы настройки различных компонентов и позволяет подготовиться к экзаменам Red Hat Certified Technician (RHCT) и Red Hat Certified Engineer (RHCE). При создании практической части было уделено внимание сбалансированности самостоятельных заданий. Они довольно разнообразны – от простых для «новичков», с подробным описанием всех шагов, до более сложных, с возможностью самостоятельного выполнения различными способами.

Курс предназначен для системных администраторов и инженеров, обладающих начальными знаниями об ОС Linux, в объеме предыдущей книги серии – «Основы работы с Linux»

Войтов Никита Михайлович

Администрирование ОС Red Hat Enterprise Linux.

Учебный курс

Главный редактор *Мовчан Д. А.*
dm@dmk-press.ru
Корректор *Сняева Г. И.*
Верстка *Паранская Н. В.*
Дизайн обложки *Мовчан А. Г.*

Подписано в печать 26.10.2010. Формат 70×100¹/₁₆.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 36. Тираж 1000 экз.
№

Web-сайт издательства: www.dmk-press.ru

© Войтов Н. М., 2011

© Softline Academy, 2011

© Оформление, издание, ДМК Пресс, 2011

ISBN 978-5-94074-677-5



Содержание

Введение	6
Условные обозначения	7
Схема класса и виртуальной сети	8
Модуль 1. Системное администрирование	10
1.1. Процесс загрузки ОС	10
1.1.1. Универсальный загрузчик GRU	11
1.1.2. Работа с командами загрузчика GRUB	14
1.2. Уровни выполнения	18
1.3. Определение характеристик оборудования и его совместимости ..	23
1.4. Установка ОС	25
1.4.1. Методы установки ОС Linux	25
1.4.2. Создание инсталляционного ресурса	26
1.4.3. Процесс установки	27
1.4.4. Автоматизированная установка kickstart	32
1.4.5. Установка с использованием технологии PXE	36
1.5. Управление пакетами программ	36
1.5.1. Менеджер пакетов RPM	37
1.5.2. Менеджер пакетов YUM	42
1.5.3. Создание пакетов RPM	44
1.6. Организация хранения данных	47
1.6.1. Создание и управление массивами RAID	48
1.6.2. Создание и управление логическими томами с помощью LVM ..	51
1.7. Резервное копирование и восстановление данных	52
1.7.1. Утилиты dump и restore	53
1.7.2. Утилита tar	54
1.7.3. Утилита rsync	55
1.8. Автоматизация выполнения задач	56
1.9. Журналирование системных событий	58
1.10. Мониторинг и оптимизация производительности	61
1.10.1. Процессор	61
1.10.2. Память	64
1.10.3. Подсистема дискового ввода-вывода	66
1.10.4. Комплексные средства мониторинга системных ресурсов	67
1.11. Установка программ из исходного кода	68

Модуль 2. Сетевое администрирование	76
2.1. Централизованное хранилище данных. Каталоги LDAP.....	76
2.1.1. Настройка конфигурационных файлов LDAP	78
2.1.2. Создание каталога LDAP	79
2.1.3. Подключение к серверу LDAP	80
2.2. Организация общего доступа к файлам (NFS, SMB)	82
2.2.1. Организация общего доступа на основе NFS	82
2.2.2. Организация общего доступа на основе Samba	86
2.3. Система разрешения имен (DNS). Автоматизация получения сетевых параметров (DHCP)	89
2.3.1. Сервис DNS	89
2.3.2. Сервис DHCP	95
2.4. Администрирование веб-сервера Apache	96
2.5. Администрирование прокси-сервера SQUID	101
2.6. Защищенное администрирование. Пакет OpenSSH.....	104
2.7. Система обмена почтовыми сообщениями. Хранилище почтовых данных	107
2.7.1. Конфигурирование Sendmail	107
2.7.2. Конфигурирование Dovecot.....	110
2.8. Администрирование общих сетевых сервисов (xinetd, FTP, NTP)	111
2.8.1. Сервис xinetd.....	111
2.8.2. Сервис FTP	112
2.8.3. Сервис NTP	114
2.9. Основы маршрутизации в сетях TCP/IP	115
Модуль 3. Организация информационной безопасности	118
3.1. Управление интеллектуальными списками доступа (SELinux)	118
3.1.1. Режим работы и политика механизма SELinux.....	119
3.1.2. Использование утилиты SELinux Troubleshooting Tool.....	120
3.1.3. Работа с контекстами безопасности	120
3.2. Организация межсетевого экрана (iptables)	121
3.3. Организация виртуальных частных сетей (VPN).....	126
3.4. Аудит системных событий (auditd)	130
Приложения	135
Приложение 1. Конфигурационный файл загрузчика GURB2	135
Приложение 2. Описание основных категорий и групп пакетов ОС Linux.....	136
Приложение 3. Основные команды управления LVM	138
Приложение 4. Основные журнальные файлы ОС Linux	140

Практические работы	142
Описание виртуальных машин	142
Практическая работа 1. Системное администрирование ОС Linux	143
Упражнение 1.1. Изменение параметров загрузки ядра ОС Linux.....	143
Упражнение 1.2. Управление системными сервисами	144
Упражнение 1.3. Сетевая инсталляция ОС Linux с использованием сервиса NFS	145
Упражнение 1.4. Установка ядра ОС Linux	148
Упражнение 1.5. Создание отказоустойчивого и масштабируемого хранилища.....	149
Упражнение 1.6. Синхронизация данных.....	152
Упражнение 1.7. Централизованное хранилище журнальных файлов..	156
Упражнение 1.8. Мониторинг системных ресурсов	157
Самостоятельные упражнения и дополнительные вопросы	159
Практическая работа 2. Сетевое администрирование ОС Linux	160
Упражнение 2.1. Сетевая аутентификация пользователей LDAP	160
Упражнение 2.2. Настройка монтирования домашнего каталога пользователя по требованию	163
Упражнение 2.3. Настройка динамического обновления записей DNS.....	165
Упражнение 2.4. Организация централизованного веб-доступа к документации ОС Linux.	169
Упражнение 2.5. Настройка аутентификации SSH по публичному ключу.	172
Упражнение 2.6. Создание защищенного почтового сервиса на основе MTA Sendmail и Dovecot.	174
Упражнение 2.7. Организация файлообменного сервиса FTP.....	178
Упражнение 2.8. Синхронизации времени.	181
Самостоятельные упражнения и дополнительные вопросы	183
Практическая работа 3. Организация информационной безопасности ОС Linux	184
Упражнение 3.1. Анализ работы механизма SELinux.....	184
Упражнение 3.2. Настройка ограничений на подключение к Telnet	185
Упражнение 3.3. Организация межсетевого экрана	186
Самостоятельные упражнения и дополнительные вопросы	188
Упражнение 3.4. Организация контроля целостности файлов.....	189
Упражнение 3.5. Организация VPN тунеля	190



Введение

Данное пособие – второе в серии книг по Linux, издаваемых совместно ДМК-пресс (www.dmk-press.ru) и Softline Academy Alliance (www.it-academy.ru). Данный курс предполагает наличие у слушателей знаний в объеме курса RH-033 «Основы работы с ОС Red Hat Enterprise Linux» и является следующей ступенью в освоении ОС Linux. Курс позволяет получить знания и навыки, необходимые для успешного администрирования ОС Linux. В качестве основного рассматриваемого в данном курсе дистрибутива используется Red Hat Enterprise Linux 5 (RHEL)¹, являющийся коммерческим решением компании Red Hat.

Курс предназначен для системных администраторов и инженеров начального² уровня знаний, стремящихся освоить системное и сетевое администрирование ОС Linux, а также использовать данную ОС для решения задач информационной безопасности. После прохождения данного курса, при желании, слушатели могут успешно сертифицироваться на звания Red Hat Certified Technician (RHCT) и Red Hat Certified Engineer (RHCE)³.

Предлагаемый вашему вниманию конспект лекций и практические работы – это основной учебный материал для проведения лекционных и практических занятий по курсу «Администрирование ОС Red Hat Enterprise Linux 5» в учебных центрах Softline Academy (www.it-academy.ru). Эти учебные центры создаются в рамках инициативы Softline Academy Alliance, цель которой объединить учебные заведения и организации, заинтересованные в качественной и эффективной подготовке студентов и молодых специалистов для работы в области ИТ.

Курс разработан преподавателями Учебного центра ВМК МГУ & Softline Academy (www.it-university.ru), который является первым в России авторизованным учебным центром программы Microsoft IT Academy и первой Академией Softline. Курс рассчитан на 40 академических часов и может быть освоен как самостоятельно, так и под руководством опытного преподавателя в любой из двадцати пяти Академий Softline, находящихся в восемнадцати регионах России.

Данный курс состоит из трех частей, изучение каждой из которых предполагает хорошее усвоение предыдущего материала.

Первая часть курса полностью посвящена системному администрированию и

¹ Здесь и далее по тексту под «ОС Linux» будет пониматься дистрибутив Red Hat Enterprise Linux 5. Начальная подготовка предполагают уверенное знание.

² Начальная подготовка предполагают уверенное знание основ ОС Linux и командного интерпретатора bash, полученных в ходе изучения курса RH-033.

³ Основная информация по программе сертификации Red Hat представлена на сайте <https://www.redhat.com/certification/>

содержит материал, необходимый для успешной сдачи экзамена RHCT (RH202). Данная часть курса содержит в себе вопросы инсталляции ОС Linux, управления пакетами программ, обеспечения резервного копирования, организации хранилища данных, а также другие важные вопросы, связанные с системным администрированием.

Вторая часть курса ориентирована на администрирование сетевых служб и содержит материал, необходимый для успешной сдачи экзамена RHCE (RH302). В данной части курса рассматриваются вопросы организации каталогов LDAP, проектирования и развертывания веб-серверов на базе Apache, управления системами кеширования и фильтрации контента, администрирования общих сетевых служб, а также использования ОС Linux для решения задач маршрутизации данных в сетях TCP/IP.

В третьей части курса рассматриваются вопросы информационной безопасности, которые должны знать специалисты, сертифицирующиеся на звание RHCE и желающие продолжить дальнейшее обучение в области защиты данных.

После завершения обучения данному курсу вы будете обладать необходимыми знаниями, и уметь:

- устанавливать, настраивать и обновлять ОС Linux;
- устанавливать и настраивать дополнительные пакеты программ;
- устанавливать программы из исходного кода;
- определять характеристики серверного оборудования и выбирать оптимальную конфигурацию ОС Linux с учетом системных требований;
- эффективно управлять системными процессами и сервисами;
- администрировать учетные записи пользователей и групп;
- обеспечивать надежное хранение и резервирование данных;
- администрировать основные сетевые службы, поддерживаемые ОС Linux;
- настраивать маршрутизацию трафика в сетях TCP/IP, используя ОС Linux и соответствующие программы;
- организовывать безопасное функционирование системных и сетевых служб, используя механизм SELINUX;
- настраивать межсетевой экран;
- настраивать аудит системных и сетевых событий;
- обеспечивать ведение централизованного журнала событий;
- определять «узкие» места в производительности ОС Linux и приложений;
- восстанавливать ОС Linux в случае программных или аппаратных сбоев.

Условные обозначения

В данном пособии применяются следующие условные обозначения.

Имена файлов и папок начинаются со строчных букв (при работе в командной строке или графической оболочке регистр букв всегда имеет значение).

Аббревиатуры напечатаны ПРОПИСНЫМИ БУКВАМИ.

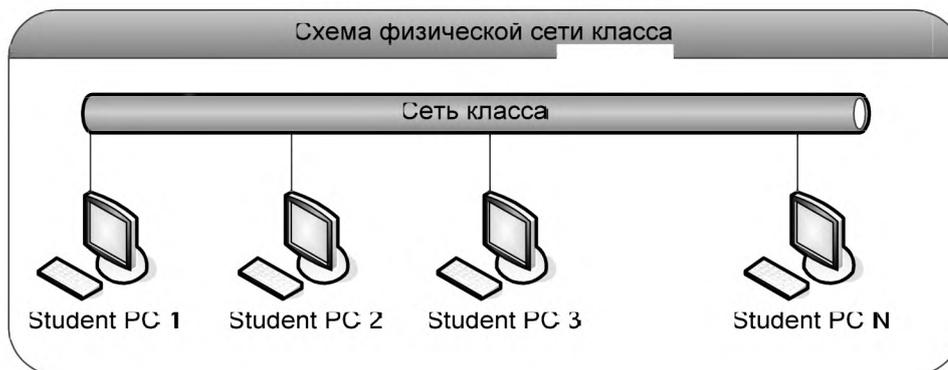
Листинги кода, примеры команд, а также текста, выводимого на экран, выделены данным шрифтом, причем ввод команды выделен **жирным шрифтом**.

Необязательные аргументы команд заключены в квадратные скобки (например: данный [аргумент] является необязательным).

Обязательные аргументы команд записываются без квадратных скобок.

Ключевые термины выделены *полужирным курсивом*.

Схема класса и виртуальные машины



Класс, в котором выполняются практические работы, состоит из физических компьютеров, объединенных в локальную сеть с адресом 192.168.1.0/24 и имеющих динамическую IP-адресацию (DHCP). Все практические работы по данному курсу выполняются на виртуальных машинах, работающих под управлением ПО VMware Player.

Учетные данные для регистрации на физическом компьютере слушателя (Host PC):

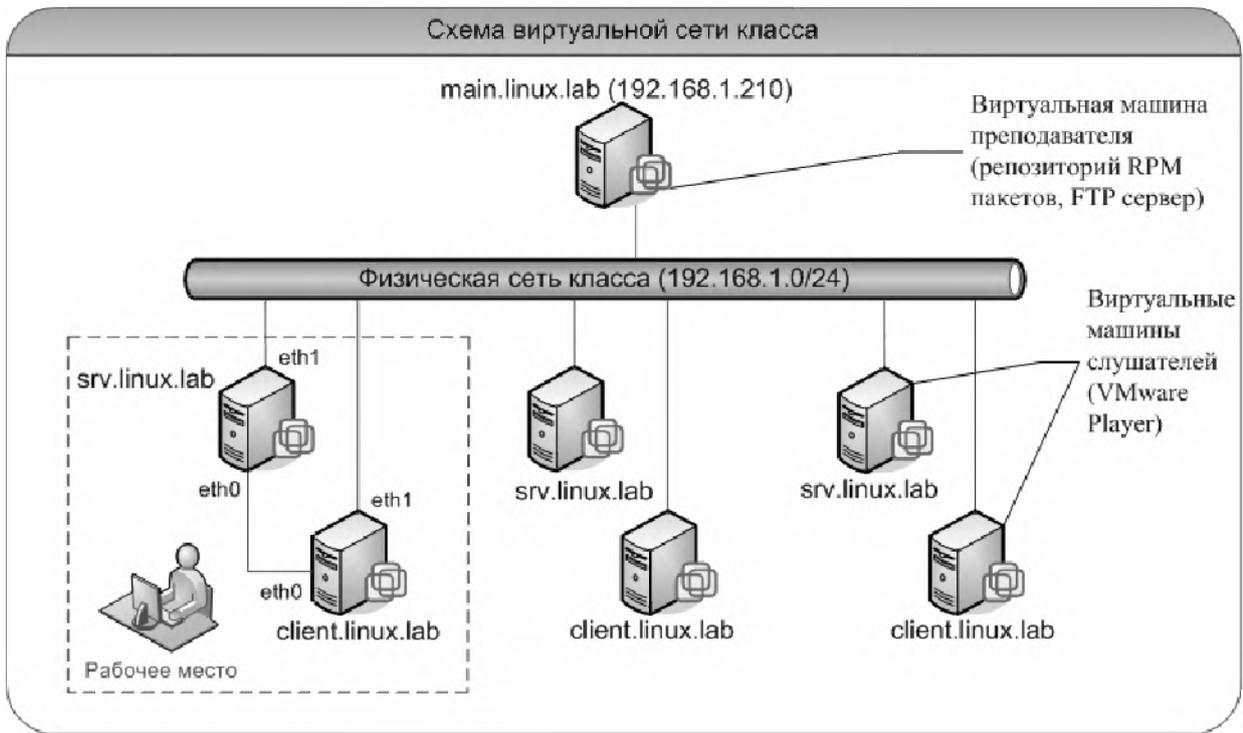
Имя пользователя: _____

Пароль: _____

Домен (если есть): _____

В качестве операционной системы *хостовой машины* (физической рабочей станции слушателя, на которой работает ПО VMware Player) используется ОС Microsoft Windows XP. В качестве операционной системы *гостевых машин*, в которых непосредственно выполняются все практические работы, используется ОС Red Hat Enterprise Linux 5 (RHEL5).

У каждого слушателя имеется одна виртуальная машина `rhel5.linux.lab`. Данная виртуальная машина имеет один сетевой интерфейс (`eth0`), используемый для связи между виртуальными машинами и выхода в физическую сеть класса. Все практические работы выполняются на виртуальной машине `rhel5.linux.lab`.





Модуль 1. Системное администрирование

Изучив данный модуль, вы научитесь:

- описывать процесс загрузки ОС Linux и настраивать загрузчик GRUB;
- описывать уровни выполнения (Runlevels);
- управлять загрузкой сервисов ОС Linux;
- описывать процедуру и варианты инсталляции ОС Linux;
- определять аппаратную конфигурацию оборудования, на котором установлена ОС Linux;
- управлять пакетами ОС Linux при помощи менеджеров пакетов **rpm** и **yum**;
- управлять дисковыми разделами, файловыми системами, RAID-группами и томами LVM;
- использовать основные методы резервного копирования данных в ОС Linux;
- автоматизировать выполнение задач при помощи планировщиков **cron** и **at**;
- настраивать журналирование системных событий;
- организовывать мониторинг системных ресурсов ОС Linux.

1.1. Процесс загрузки ОС

После включения компьютера происходит выполнение специального кода начальной загрузки. На персональных компьютерах данный код представлен в виде базовой подсистемы ввода/вывода – **BIOS** (Basic Input/Output System). Коду BIOS известно почти все об устройствах, расположенных на материнской плате: о контроллере жестких дисков, плате сетевого адаптера, контроллере клавиатуры и других устройствах. В настройках BIOS можно выбирать, с какого устройства следует начинать загрузку ОС. После того как подсистема BIOS определила, с какого устройства ей следует загружать ОС, считываются первые 512 байт данного устройства. Эта часть диска называется **загрузочной записью (MBR – Master Boot Record)**. В данной записи хранится программа **первичной загрузки (stage1)**, которая сообщает компьютеру о том, в каком разделе диска расположена программа вторичной загрузки (**stage2 – непосредственный загрузчик ОС**). В данном курсе в качестве такого загрузчика рассматривается **GRUB (GRand Unified Boot Loader)**. Помимо программы вторичной загрузки существует еще дополнитель-

ная программа (stage1.5)¹, помогающая загрузчику ОС определить некоторые параметры файловой системы. Это дополнительная программа используется в том случае, если программа вторичной загрузки не была установлена непосредственно в загрузочную запись MBR.

После того, как управление передается программе вторичной загрузки ОС (stage2), загрузчик GRUB приступает к своей основной обязанности – загрузке ОС. Если среди опций загрузки ядра отсутствует параметр **quiet**, то в процессе загрузки система будет отображать служебную информацию на экран. Данную служебную информацию можно просмотреть не только в процессе загрузки ОС, но и используя команду **dmesg** или файл **/var/log/dmesg**.

Вывод служебной информации существенно зависит от оборудования, на котором установлена ОС Linux. В данной информации можно выделить следующие основные показатели:

- версия ядра ОС;
- количество распознанной оперативной памяти;
- количество процессоров (**CPU0**, **CPU1** и т.д.);
- статус режима работы механизма **SELinux**²;
- параметры загрузки ядра ОС;
- количество свободной памяти отданной под начальный диск памяти (**initramfs**);
- жесткие диски и соответствующие им разделы (например, **/dev/sda**);
- сетевые платы (например, **eth0**);
- смонтированные файловые системы (например, **ext3**);
- разделы подкачки и их размер.

Анализ данной информации может помочь в случае возникновения проблем загрузки ОС Linux.

1.1.1. Универсальный загрузчик GRUB

Одним из вариантов загрузчика ОС Linux является программа **GRUB**. После того как программа, хранящаяся в загрузочной записи, сообщит компьютеру, что загрузчик ОС находится в разделе **/boot**, управление передается загрузчику ОС и на экран выводится его основное меню. Меню загрузчика GRUB позволяет настраивать различные варианты загрузки ядра ОС Linux, выбирать операционную систему, которая должна быть загружена, а также выполнять операции по восстановлению системы, используя собственный интерфейс командной строки. Основные команды, которые используются для управления загрузчиком GRUB, приведены табл. 1.1.

1 В версии загрузчика GRUB 2 данная программа отсутствует ввиду наличия собственного режима восстановления (rescue mode).

2 Механизм **SELinux** используется для контроля доступа приложения к ресурсам ОС.

Таблица 1.1. Основные команды работы в меню загрузчика GRUB

Команда	Описание
a	Используется для изменения параметров загрузки ядра.
b	Используется для загрузки ОС (выделенная запись меню GRUB).
c	Используется для входа на интерфейс командной строки.
d	Используется для удаления текущей записи меню GRUB.
e	Используется для редактирования меню GRUB.
o	Используется для вставки пустой строки под текущей записью в меню GRUB (режим редактирования меню GRUB).
O	Используется для вставки пустой строки над текущей записью в меню GRUB (режим редактирования меню GRUB).

Параметры меню загрузчика GRUB записываются в файл `/boot/grub/menu.lst`, который можно отредактировать после окончания процесса загрузки ОС. По умолчанию конфигурация загрузчика GRUB содержится в файле `/boot/grub/grub.conf`, однако в каталоге `/etc` существует символическая ссылка `/etc/grub.conf` на данный файл, так что редактирование конфигурации загрузчика возможно и через файл `/boot/grub/grub.conf`, и через ссылку `/etc/grub.conf`.

Загрузчик GRUB является универсальным загрузчиком и позволяет загружать не только ОС Linux, но и другие ОС, включая Windows, используя вызов сторонних загрузчиков.

Для изменения параметров³ загрузки ядра используется команда **a**, присутствующая в меню GRUB. Например, если в конец строки загрузки ядра добавить параметр **single**, то система начнет загружаться в режиме **одного пользователя** (single-user). В случае если ОС некорректно распознает количество установленной оперативной памяти, необходимо добавить параметр **mem=XМ**, где X обозначает количество мегабайт оперативной памяти.

В случае возникновения проблем с загрузкой ОС Linux в первую очередь необходимо проверить конфигурацию загрузчика GRUB, содержащуюся в файле `/boot/grub/grub.conf`. В следующем листинге приведено типовое содержание данного конфигурационного файла.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this
file
# NOTICE:  You have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/sda2
#           initrd /initrd-version.img
#boot=/dev/sda
default=0
```

³ Полный список параметров ядра ОС Linux содержится по адресу <http://www.kernel.org/doc/Documentation/kernel-parameters.txt>

```
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-53.el5)
    root (hd0,0)
    kernel /vmlinuz-2.6.18-53.el5 ro root=LABEL=/ rhgb quiet
    initrd /initrd-2.6.18-53.el5.img
```

В первой строке файла сообщается о том, что он был создан программой **Anaconda**, являющейся штатной программой-установщиком ОС Linux. В следующей строке содержится предупреждение, напоминающее о том, что изменения, сделанные в файле **grub.conf**, не будут записаны в загрузочную запись MBR, поскольку данный файл автоматически привязан к программе первичной загрузки stage1, расположенной в загрузочной записи MBR.

Строка со словом Notice появляется в том случае, если каталог **/boot** был сделан отдельным разделом. Каталог **/boot** ассоциирован с параметром **root**, в данном случае **root (hd0,0)**. Однако на самом деле корневой каталог **/** находится во втором разделе устройства **sda** и имеет абсолютный путь устройства **/dev/sda2**. Загрузчик GRUB использует свои правила именования физических устройств, отличающиеся от стандартных соглашений, принятых в ОС Linux. Имя устройства в загрузчике GRUB выглядит, например, так: **(hd0,0)**. Первое числовое значение определяет номер физического диска (нумерация ведется с нуля), а второе – номер раздела (правила нумерации аналогичны). В рассматриваемом листинге запись **(hd0,0)** соответствует первому разделу первого жесткого диска – **/dev/sda1**. К данному устройству монтируется каталог **/boot**, содержащий файл ядра ОС, образ памяти **initrd** и конфигурационные файлы загрузчика GRUB. Последняя закомментированная строка файла **grub.conf** сообщает, что загрузочным устройством, содержащим загрузочную запись MBR, является устройство **/dev/sda**.

После закомментированных строк записаны непосредственно команды загрузчика GRUB. Первая команда **default**, обозначает номер заголовка меню (**title**), который необходимо использовать для загрузки по умолчанию. Нумерация заголовков меню начинается с 0. В листинге присутствует лишь один заголовок **title**, при выборе которого будет осуществлена загрузка ОС **Red Hat Enterprise Linux Server (2.6.18-53.el5)**. Команда **timeout** обозначает время задержки в секундах, по истечении которого автоматически будет загружена ОС, указанная в директиве **default**. Директива **splashimage** указывает на файл, содержащий графические примитивы для отображения меню. Директива **hiddenmenu** обозначает, что по умолчанию меню скрыто от пользователя и вместо него на экране отображается строка вида **Booting Red Hat Enterprise Linux Server (2.6.18-53.el5) in 5 seconds...** Далее следуют команды описания расположения корневого раздела **/**, в данном случае это **(hd0,0)**, расположения файла ядра ОС Linux **vmlinuz-2.6.18-53.el5**, с указанием соответствующих параметров загрузки ядра, расположения файла **начального образа памяти** **initrd-2.6.18-53.el5.img**. Начальный образ памяти (Initial RAM disk) в процессе загрузки ОС создает временную файловую систему, содержащую модули ядра и программы, необходимые для монтирования файловых систем и запуска всех остальных системных процессов.

Параметры загрузки ядра указываются через пробел. По умолчанию используются следующие параметры **ro root=LABEL=/ rhgb quiet**.

Первый параметр сообщает о том, что каталог **/boot** открывается в режиме чтения: это делается с целью защиты раздела от случайной записи неверных данных. Второй параметр обозначает, что корневой каталог ассоциирован с меткой **/(root=LABEL=/)**. Последние два параметра **rhgb** и **quiet** указывают на то, что, во-первых, следует использовать графическое меню загрузчика GRUB и, во-вторых, не отображать служебные сообщения в процессе загрузки ОС.

1.1.2. Работа с командами загрузчика GRUB

Загрузчик GRUB имеет собственный набор команд, предназначенный для его настройки. Запустить интерфейс для ввода данных команд можно из меню загрузчика, выполнив команду **c**, или непосредственно из командного интерпретатора, выполнив команду **grub**. Для просмотра всех возможных команд загрузчика GRUB используется клавиша **<TAB>**. Командный интерфейс загрузчика GRUB имеет базовые возможности, присутствующие в командном интерпретаторе **bash**; в частности, можно использовать автодополнение ввода и историю введенных команд.

В следующем листинге приведены примеры использования команд загрузчика GRUB.

```
grub> find /grub/stage1
(hd0,0)
grub> root (hd0,<TAB>
Possible partitions are:
  Partition num: 0,  Filesystem type is ext2fs, partition type 0x83
  Partition num: 1,  Filesystem type is ext2fs, partition type 0x83
  Partition num: 2,  Filesystem type unknown, partition type 0x82
  Partition num: 4,  Filesystem type unknown, partition type 0x8e
grub> root (hd0,1)
Filesystem type is ext2fs, partition type 0x83
grub> root
(hd0,1): Filesystem type is ext2fs, partition type 0x83
grub> find (hd0,0)/grub/grub.conf
(hd0,0)
(hd0,1)
grub> cat (hd0,1)/etc/redhat-release
Red Hat Enterprise Linux Server release 5.1 (Tikanga)
```

В первой строке листинга используется команда **find** для поиска устройства, содержащего программу первичной загрузки **stage1**. Далее при помощи команды **root** осуществляется поиск корневого раздела **/** (команда **root (hd0,<TAB>)**), его ассоциация с физическим устройством **(hd0,1)** (команда **root (hd0,1)**) и вывода типа его файловой системы⁴ (команда **root**). В предпоследней команде осуществляется поиск физических устройств, которые содержат файл **grub.conf**. Следует заметить, что в выводе данной команды содержится два устройства **(hd0,0)** и

⁴ Большинство типов файловых систем, записанных в шестнадцатеричном формате, можно посмотреть по адресу <http://www.essdatarecovery.com/hexcodes.asp> или при помощи команды **fdisk**.

(hd0,1), соответствующие разделам **/boot** и **/**, поскольку файл **grub.conf** содержится в обоих разделах. Последняя команда используется для вывода содержимого текстового файла **/etc/redhat-release**, содержащего информацию о релизе ОС Linux.

Описанные выше особенности отчасти применимы к новой версии загрузчика GRUB – **GRUB2**, имеющей модульную архитектуру и обладающей большей переносимостью. Загрузчик **GRUB2** в настоящее время находится в активной разработке и имеет следующие особенности:

- Поддержка скриптования (циклы, переменные, функции);
- Динамическая загрузка модулей с жесткого диска при необходимости (RAID, LVM);
- Интернационализация (отображение не ASCII-символов);
- Собственный командный режим восстановления;
- Графический интерфейс.

Наиболее важные отличия **GRUB2** от предыдущей версии (**GRUB legacy**) следующие:

- Настройка меню загрузчика выполняется через файл **grub.cfg** (файл **menu.lst** отсутствует);
- Нумерация разделов начинается с 1 (**(hd0,1)** – соответствует первому разделу на жестком диске);
- Добавлены дополнительные команды⁵ и изменены названия некоторых стандартных команд предыдущей версии.

Загрузчик **GRUB2** имеет следующие конфигурационные файлы и каталоги:

- **/etc/default/grub**⁶ – позволяет устанавливать глобальные параметры загрузчика (например, при указании параметра **GRUB_CMDLINE_LINUX=acpi=off**, ко всем командам «**linux**» в файле **grub.cfg** будет добавлен параметр **acpi=off**).
- **/etc/grub.d** – содержит набор сценариев командной оболочки, необходимых для генерации файла **grub.cfg**. Здесь присутствуют сценарии, помогающие определить имеющиеся операционные системы, а также добавить собственные сценарии пользователя в меню загрузчика. Порядок представления результатов сценариев в меню загрузчика определяется первыми числовыми символами в имени файла сценария (например, результат выполнения сценария **00_header**, будет присутствовать в меню первым).
- **/boot/grub/grub.cfg** – основной конфигурационный файл, содержащий информацию о загружаемых системах.

Для создания конфигурационного файла **grub.cfg** используется команда **grub-mkconfig**:

⁵ Сравнение команд загрузчиков GRUB2 и GRUB legacy представлено здесь: <http://grub.enbug.org/CommandList>

⁶ Дополнительные параметры файла **/etc/default/grub** описаны в приложении 1

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

В результате выполнения данной команды в каталоге `/boot/grub` будет создан файл `grub.cfg` примерно следующего содержания:

```
### BEGIN /etc/grub.d/00_header ###
set default=0
set timeout=5
set root=(hd0,3)
search --fs-uuid --set alae9df0-b8d3-4fe2-ad81-75a3c2f03c5d
if font /usr/share/grub/ascii.pff ; then
  set gfxmode=640x480
  insmod gfxterm
  insmod vbe
  terminal gfxterm
fi
### END /etc/grub.d/00_header ###
### BEGIN /etc/grub.d/05_debian_theme ###
set menu_color_normal=cyan/blue
set menu_color_highlight=white/blue
### END /etc/grub.d/05_debian_theme ###
### BEGIN /etc/grub.d/10_hurd ###
### END /etc/grub.d/10_hurd ###
### BEGIN /etc/grub.d/10_linux ###
menuentry "Debian GNU/Linux, linux 2.6.26-2-686" {
  set root=(hd0,2)
  search --fs-uuid --set dc47f706-7cee-4b03-8bd0-5efbf1996d9f
  linux /vmlinuz-2.6.26-2-686 root=UUID=alae9df0-b8d3-4fe2-ad81-
75a3c2f03c5d ro
  initrd /initrd.img-2.6.26-2-686
}
menuentry "Debian GNU/Linux, linux 2.6.26-2-686 (single-user mode)" {
  set root=(hd0,2)
  search --fs-uuid --set dc47f706-7cee-4b03-8bd0-5efbf1996d9f
  linux /vmlinuz-2.6.26-2-686 root=UUID=alae9df0-b8d3-4fe2-ad81-
75a3c2f03c5d ro single
  initrd /initrd.img-2.6.26-2-686
}
### END /etc/grub.d/10_linux ###
```

Как видно из листинга, формат записи изменился существенно: для задания параметров меню и указания загрузочного раздела (**root**), содержащего образ ядра и модули **GRUB**, используется команда **set**. Для поиска файловой системы, содержащей образ ядра, используется команда **search** (взамен команды **find**). Для загрузки ядра ОС в оперативную память используются команды **linux** или **multiboot** (замена команды **root**). Нетрудно заметить, что в конфигурационном файле присутствует оператор **if**, содержащий набор команд, загружающий встроенный терминал загрузчика **GRUB2** – **gfxterm**. Параметр **--fs-uuid** определяет уникальный идентификатор **UUID**⁷ загрузочного раздела.

⁷ **Universally Unique Identifier (UUID)** используется для уникальной идентификации различной информации. Например, в случае использования **UUID** для устройств, перенос данных на другое устройство или смена номера устройства (LUN) не повлияет на корректную идентификацию загрузочного раздела в процессе загрузки ОС.

В случае изменения конфигурационных файлов в каталоге `/etc/grub.d` или в файле `/etc/default/grub`, следует выполнить команду `update-grub2` для обновления файла меню загрузчика. Данная команда на самом деле вызывает команду `grub-mkconfig -o /boot/grub/grub.cfg`, которая выполняет несколько специальных сценариев и формирует файл `/boot/grub/grub.cfg`.

Как было сказано ранее основная особенность загрузчика **GRUB2** – модульная архитектура. Для загрузки необходимых модулей используется команда `insmod <название_модуля>`. Например, если раздел `/boot` расположен поверх **LVM** тома, то в описании пункта меню в файле `grub.cfg` необходимо добавить команду загрузки модуля `lv`:

```
insmod lv
```

и указать загрузочный раздел, используя следующую нотацию:

```
set root=(lvm_группа-lvm_том)
```

Важной особенностью загрузчика **GRUB2** является наличие собственного режима восстановления `rescue mode`, использующегося в случае возникновения проблем загрузки основной части загрузчика – программы `stage2`. Если загрузчик не сможет считать параметры меню из файла `grub.cfg`, но сможет определить системные диски, пользователь попадает в нормальный командный режим, отличающийся следующим приглашением:

```
sh:grub>
```

Если в системе присутствуют более серьезные проблемы, то загружается режим восстановления со следующим приглашением:

```
grub rescue>
```

Режим **rescue mode** является, по сути, усеченным нормальным режимом с ограниченным набором команд. Прежде чем пытаться что-то восстанавливать стоит попробовать загрузить нормальный режим, используя соответствующий модуль и переменную `prefix`, хранящуюся в **MBR** и содержащую расположение основной программы `stage2` и модулей:

```
grub rescue> set prefix=(hdX,Y)/grub
grub rescue> insmod (hdX,Y)/grub/normal.mod
rescue:grub> normal
```

В данном листинге `X` – соответствует номеру устройства, а `Y` – соответствует номеру раздела. Для расширения возможностей консоли необходимо загрузить модуль `linux.mod`:

```
grub rescue> insmod (hdX,Y)/grub/linux.mod
```

После этого станет доступна стандартная консоль **GRUB2**, в которой следует выполнить дальнейшую загрузку ОС:

```
set root=(hdX,Y)
linux /boot/vmlinuz-2.6.18-53.el5 ro root=LABEL=/
initrd /initrd-2.6.18-53.1.21.el5.img
boot
```

Данная последовательность команд стандартная в большинстве случаев: сначала определяется загрузочный раздел, содержащий образы ядра и *ram*-диска **initrd**, затем выполняется последовательная загрузка данных образов в память с указанием необходимых параметров и, наконец, дальнейшее управление процессом загрузки передается ядру при помощи команды **boot**.

1.2. Уровни выполнения

После того, как ядро было загружено загрузчиком **GRUB**, оно выполняет базовую настройку необходимых устройств, в том числе загрузку *ram*-диска **initrd**, содержащего драйверы устройств и позволяющего смонтировать корневой раздел. Затем загрузка ОС Linux продолжается вызовом процесса **init**. Данный процесс запускает скрипт `/etc/rc.d/rc.sysinit`, который осуществляет загрузку дополнительных модулей ядра (например, драйвера поддержки программных RAID-массивов), проверку файловой системы корневого раздела и монтирование корневого раздела в режиме чтения/записи. Далее процесс **init** на основании данных, содержащихся в файле `/etc/inittab`, определяет, какой уровень выполнения (*runlevel*) необходимо использовать, и последовательно продвигается от уровня 0 к уровню по умолчанию, указанному в файле `/etc/inittab` директивой `id:5:initdefault:.`

В файле `/etc/inittab` содержится информация обо всех уровнях выполнения, о виртуальных терминалах, загрузочных сценариях и других системных параметрах. **Уровнями выполнения** (*runlevels*) в ОС Linux называются несколько специальных режимов работы, на каждом из которых происходит запуск или остановка определенных процессов-демонов. Описание уровней выполнения ОС Linux приведено в табл. 1.2.

Таблица 1.2. Уровни выполнения ОС Linux

Уровень	Значение
0	Остановка системы.
1	Запуск ОС в режиме одного пользователя (<i>single-user mode</i>) без поддержки сетевых сервисов. Загрузка ОС и последующая работа производятся в режиме командной строки.
2	Запуск ОС в многопользовательском режиме без поддержки сетевых сервисов. Загрузка ОС и последующая работа производятся в режиме командной строки.
3	Запуск ОС в многопользовательском режиме с поддержкой сетевых сервисов. Загрузка ОС и последующая работа производятся в режиме командной строки.
4	Не используется.
5	Запуск ОС в многопользовательском режиме с поддержкой сетевых сервисов. Загрузка ОС и последующая работа производятся в графическом режиме.
6	Перезапуск системы.

Смена уровней выполнения скриптом `/etc/rc.d/rc`, который запускает все остальные системные процессы, посредством обращения к символьным ссылкам `init`-скриптов, содержащихся в каталогах `rc0.d` - `rc6.d`. Имена ссылок начинаются с префикса **S** или **K**, за которым следует порядковый номер и имя процесса-демона, управляемого данным `init`-скриптом (например, **S55sshd**). При переходе на заданный уровень выполнения, скрипт `rc` выполняет все скрипты с префиксом **S** в порядке возрастания порядковых номеров, причем выполнение скриптов осуществляется с параметром `start`. При выходе с текущего уровня выполняются все скрипты с префиксом **K** в порядке возрастания порядковых номеров, причем выполнение скриптов осуществляется с параметром `stop`. Например, при переходе с 3-го на 5-й уровень выполняются все скрипты каталога `/etc/rc.d/rc5.d` с префиксом **K**, а затем выполняются все скрипты с префиксом **S** того же каталога.

В следующем листинге приведен частичный вывод каталога `/etc/rc.d/rc3.d`, который соответствует третьему функциональному уровню загрузки. Из данного листинга видно, что демон `sshd` запускается под номером **55** после того, как все предыдущие сервисы, содержащие префикс **S**, будут запущены.

```
lrwxrwxrwx 1 root root 16 Dec  9 17:19 K02dhcdbd -> ../init.d/dhcdbd
lrwxrwxrwx 1 root root 14 Dec  9 17:21 S26hidd -> ../init.d/hidd
lrwxrwxrwx 1 root root 16 Dec  9 17:22 S28autofs -> ../init.d/autofs
lrwxrwxrwx 1 root root 15 Dec  9 17:21 S44acpid -> ../init.d/acpid
lrwxrwxrwx 1 root root 15 Dec  9 17:21 S50hplip -> ../init.d/hplip
lrwxrwxrwx 1 root root 14 Dec  9 17:20 S55sshd -> ../init.d/sshd
```

Часть строк файла `/etc/inittab` представлена в следующем листинге.

```
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
10:0:wait:/etc/rc.d/rc 0
. . . . .
16:6:wait:/etc/rc.d/rc 6
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
1:2345:respawn:/sbin/mingetty tty1
. . . . .
6:2345:respawn:/sbin/mingetty tty6
x:5:respawn:/etc/X11/prefdm -nodaemon
```

В первой строке содержится директива определения функционального уровня по умолчанию (`initdefault`). Из данного листинга видно, что по умолчанию система функционирует на уровне 5, который ассоциирован с работой в графическом режиме. Каждая строка файла `/etc/inittab` записывается в определенном формате и состоит из четырех полей:

идентификатор : уровень : действие : команда

- **идентификатор** – поле, состоящее из 1-4 символов, обозначающих функционал данной записи.
- **уровень** – поле, содержащее список уровней, для которых применяется

указанная команда. Например, запись 345 означает выполнение команды на уровнях 3, 4 и 5.

- **действие** – специальное действие, которое должен выполнить процесс **init** прежде чем перейти к считыванию следующей строки файла **/etc/inittab**. Например, запись **wait** говорит о том, что процесс **init** должен дождаться завершения выполнения указанной команды. Запись **respawn** говорит о том, что процесс **init** должен повторно выполнить команду в случае ее завершения.
- **команда** – в данном поле указана команда, которую необходимо выполнить, со всеми ее аргументами и опциями.

Помимо определения уровней выполнения, в файле **/etc/inittab** содержится определение ряда дополнительных команд и виртуальных терминалов. Первая из команд определяет, каким образом в системе будет осуществляться перехват клавиш **<CTRL+ALT+DELETE>**:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Следующие две команды определяют поведение системы в случае получения аварийного сигнала **powerfail (SIGPWR)**:

```
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
```

Далее в файле **/etc/inittab** следует определение виртуальных терминалов. **Виртуальный терминал** – это сеанс командной строки, в котором выполняется основная работа с системой. По умолчанию, в ОС Linux определено шесть виртуальных терминалов, однако это количество может быть увеличено до 12. Поскольку ОС Linux является многопользовательской системой, несколько разных пользователей могут одновременно работать из нескольких терминалов, включая графическую оболочку.

Переключение между терминалами осуществляется при помощи клавиш **<ALT+Fn>**, где **n** обозначает номер виртуального терминала (от 1 до 12). Например, для переключения на третий виртуальный терминал необходимо нажать **<ALT+F3>**.

Последним звеном процесса загрузки ОС Linux является скрипт **/etc/rc.d/rc.local**, который предназначен для добавления пользовательских сценариев запуска.

Управление уровнями выполнения является очень важным аспектом администрирования ОС Linux. Вы должны уметь настраивать уровень выполнения, используемый системой по умолчанию, а также определять, какие из системных процессов должны запускаться автоматически на данном уровне.

По умолчанию, если при инсталляции была установлена графическая оболочка, ОС Linux загружается на 5-й уровень выполнения, о чем свидетельствует запись **id:5:initdefault:** файла **/etc/inittab**. Для изменения уровня по умолчанию необходимо изменить второе поле данной записи. Сделанные измене-

ния вступят в силу только после перезагрузки системы. Для того чтобы изменить текущий уровень выполнения без перезагрузки системы необходимо запустить следующую команду от пользователя **root**:

```
init <уровень>
```

Параметр **уровень** задается числом от 0 до 6.

Для настройки автозапуска процессов в ОС Linux используются следующие команды и утилиты:

- chkconfig;
- ntsysv;
- service configuration tool.

Команда **chkconfig** может использоваться как для просмотра текущей конфигурации уровней выполнения, так и для их настройки и предоставляет наиболее простой механизм осуществления следующих действий:

- просмотр текущих настроек сервисов;
- изменение текущих настроек сервисов;
- добавление/удаление сервисов из структуры каталогов **/etc/rc.d**;
- включение/отключение автоматического запуска сервисов.

Команда **chkconfig** имеет следующий синтаксис:

```
chkconfig --list [имя сервиса]
chkconfig --add <имя сервиса>
chkconfig --del <имя сервиса>
chkconfig [--level <уровни>] <имя сервиса> <on|off|reset>
```

Необязательный параметр **имя сервиса** обозначает название системного процесса или сервиса, сценарий которого присутствует в каталоге **/etc/rc.d/init.d**, причем имя сервиса должно совпадать с названием соответствующего сценария.

Опция **--list** используется для просмотра текущих настроек сервиса.

Опции **--add** и **--del** используются для добавления и удаления символических ссылок сервисов в структуру каталогов **/etc/rc.d**.

Опция **--level** определяет, для каких уровней выполнения необходимо выполнить указанные далее действия.

Параметры **on**, **off**, **reset** используются для изменения статуса запуска сервиса.

При указании параметра **off** сервис исключается из автозапуска в процессе загрузки ОС. При указании параметра **on** сервис запускается в процессе загрузки ОС. При указании параметра **reset** статус запуска сервиса устанавливается согласно изначальной конфигурации, указанной в файле сценария.

В следующем листинге приведены примеры использования команды **chkconfig** для управления сервисами.

```
# chkconfig --list|grep sendmail
sendmail      0:off  1:off  2:on   3:on   4:on   5:on   6:off
# chkconfig --level 245 sendmail off
```

```
# chkconfig --list|grep sendmail
sendmail          0:off  1:off  2:off  3:on   4:off  5:off  6:off
# chkconfig sendmail reset
# chkconfig --list|grep sendmail
sendmail          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

В первой строке осуществляется вывод текущих настроек запуска сервиса **sendmail**, из которых видно, что данный сервис запускается на уровнях **2, 3, 4** и **5**. Во второй строке осуществляется отключение автозапуска сервиса **sendmail** на уровнях **2, 4** и **5**. Далее при помощи параметра **reset** осуществляется возврат первоначальных настроек автозапуска сервиса **sendmail** согласно записи (`# chkconfig: 2345 80 30`), указанной в файле сценария `/etc/rc.d/init.d/sendmail`.

Для добавления нового сервиса в структуру каталогов `/etc/rc.d` необходимо, чтобы файл сценария сервиса имел как минимум две обязательные записи:

1. `# chkconfig: 2345 80 30`

Числовое значение, следующее за символом «:», говорит о том, что в каталогах `/etc/rc.d/rc2.d`, `/etc/rc.d/rc3.d`, `/etc/rc.d/rc4.d` и `/etc/rc.d/rc5.d` (уровнях **2, 3, 4** и **5**) необходимо создать символьные ссылки на данный сценарий, причем, ссылкам с префиксом **S** будет присвоен порядковый номер **80**, а ссылкам с префиксом **K** – порядковый номер **30**.

2. `# description: <текст>`

В данной записи дается формальное описание сервиса.

Например, для сервиса **sendmail** записи, считываемые утилитой **chkconfig**, имеют следующий вид:

```
# chkconfig: 2345 80 30
# description: Sendmail is a Mail Transport Agent, which is the program \
# that moves mail from one machine to another.
```

После создания файла сценария сервиса, содержащего записи **chkconfig**, его необходимо скопировать в каталог `/etc/init.d` и сделать его исполняемым. Затем, используя команду **chkconfig --add <название_сценария>**, создать символьные ссылки в соответствующих каталогах `/etc/rc.d/rc*.d`.

Для управления автозапуском сервисов в интерактивном режиме используется команда **ntsysv**, имеющая следующий синтаксис:

```
ntsysv [--back] [--level <уровни>]
```

Для указания уровней выполнения, на которых необходимо настроить автозапуск сервисов, используется необязательная опция **--level**. В случае запуска данной команды без параметров на терминале будет отображено окно настройки сервисов для текущего функционального уровня.

Для управления сервисами в графическом режиме используется программа **service configuration tool**. Данную программу можно запустить непосредственно из командной строки, выполнив команду **system-config-services**, или из графической оболочки, выбрав ярлык **Services**, расположенный на вкладке **System** → **Administration** → **Services**.

1.3. Определение характеристик оборудования и его совместимости

Администратор должен уметь анализировать аппаратную конфигурацию сервера с уже установленной системой с целью оптимизации производительности ОС Linux, а также решать проблемы, связанные с отказом аппаратных устройств. Перед установкой ОС Linux администратор также должен проанализировать оборудование на предмет его совместимости с ОС Linux.

В сети Интернет существует большое количество ресурсов, где можно выбрать оборудование, поддерживаемое ОС Linux. Пожалуй, самым главным из этих ресурсов является проект Linux Documentation Project (LDP, <http://tldp.org/HOWTO/Hardware-HOWTO/>) и Red Hat Hardware Compatibility List (HCL, <https://hardware.redhat.com/>), предоставляемый компанией Red Hat. При выборе оборудования под ОС Linux следует обязательно просмотреть базу RHCL на наличие протестированной схожей конфигурации оборудования. Если данное оборудование официально поддерживается вендором, то можно не беспокоиться, что в процессе установки ОС Linux появятся проблемы, связанные с совместимостью. Если же проблемы все таки возникнут, следует ознакомиться с информацией, представленной на ресурсе LDP, где даются рекомендации по настройкам параметров ядра или драйверов для различного типа оборудования. Более того, если официально вендор не поддерживает установку ОС Linux на своем оборудовании, а вам необходимо установить ее на данном оборудовании, с большой вероятностью вам удастся это сделать.

Таблица 1.3. Важные файлы каталога `/proc`

Файл	Содержащаяся информация
<code>cpuinfo</code>	Установленные процессоры
<code>meminfo</code>	Количество установленной памяти
<code>devices</code>	Установленные в системе устройства
<code>interrupts</code>	Прерывания
<code>iomem</code>	Используемые адресные пространства
<code>modules</code>	Используемые в настоящее время модули ядра

В основе управления аппаратными устройствами компьютера лежит система **Basic Input Output System (BIOS)**, которая предоставляет фундаментальные сервисы ввода/вывода для операционной системы и различных ее приложений. Для предоставления данных сервисов BIOS использует следующие ресурсы устройств.

- **Запросы прерывания процессора (IRQ)**⁸. Для определения прерываний,

⁸ **Запросом на прерывание (IRQ)** называется посылка специального сигнала центральному вычислительному процессору, для того, чтобы он прервался и осуществил обработку некоторого события, например ввода данных с клавиатуры или сбор фрагментированного пакета, пришедшего на сетевой адаптер.

присутствующих в системе, можно просмотреть файл `/proc/interrupts`. Содержащаяся в нем информация может оказаться полезной, если установленное устройство не определилось или произошел конфликт двух устройств.

- **Адреса ввода/вывода (I/O)**⁹. Для определения адресов ввода/вывода используется файл `/proc/ioports`. Если в выводе данной команды обнаружены конфликты адресов, необходимо изменить аппаратную конфигурацию устройства или настройки BIOS.
- **Каналы прямого доступа к памяти (DMA)**. Они предназначены для непосредственного доступа к ресурсам памяти, минуя процессор. Для отображения используемых в системе каналов DMA необходимо просмотреть файл `/proc/dma`.

Для просмотра детальных характеристик оборудования, таких как используемый драйвер или версия микропрограммы, в ОС Linux существует ряд утилит, названия которых начинаются с символов `ls` (табл. 1.4.).

Таблица 1.4. Утилиты просмотра характеристик устройств

Файл	Описание
<code>lsusb</code>	Выводит информацию по найденным USB устройствам
<code>lspci</code>	Выводит информацию по найденным PCI устройствам
<code>lsptmci</code>	Выводит информацию по найденным PCMCIA устройствам
<code>lshal</code>	Выводит информацию по уровню аппаратных абстракций (HAL)

При запуске утилит, представленных в табл. 3.2, возможно управлять степенью детализации выводимой на экран информации, используя ключ `-v`, причем символ `v` может повторяться несколько раз.

Ядро ОС Linux использует BIOS для сбора информации об установленном оборудовании и после своего запуска практически не обращается к BIOS. Однако иногда может понадобиться запросить BIOS, не перезагружая компьютер. Для этого в ОС Linux существует пакет утилит `dmidecode`, кратко описанных в табл. 1.5.

Таблица 1.5. Утилиты просмотра информации BIOS

Файл	Описание
<code>dmidecode</code>	Используется для наглядного отображения таблицы DMI
<code>biosdecode</code>	Используется для наглядного отображения структур BIOS, хранящихся в памяти
<code>ownership</code>	Используется только в компьютерах Compaq
<code>vpddecode</code>	Используется только в компьютерах IBM

⁹ **Адресом ввода/вывода (I/O)** называется уникальная область памяти, зарезервированная для взаимодействия процессора с каким-либо устройством. Как правило, такие области памяти, предназначенные, например, для двух разных устройств, не должны пересекаться.

1.4. Установка ОС

После завершения изучения данного материала вы сможете устанавливать ОС Linux несколькими поддерживаемыми методами.

Программа-установщик ОС Linux **Anaconda** имеет достаточно простой графический интерфейс и обладает универсальными возможностями, начиная от установки системы в интерактивном режиме с внешнего носителя на единичные хосты и заканчивая автоматизированной установкой по сети с заранее определенной конфигурацией ОС на несколько хостов. Процесс установки может быть даже запланирован и настроен через сеть **RHN**¹⁰ – специальный сервис удаленного управления и поддержки компании Red Hat. Все эти методы установки ОС могут использовать единый *инсталляционный ресурс*¹¹.

1.4.1. Методы установки ОС Linux

Важно понимать, что не существует наилучшего универсального метода установки ОС Linux: выбор метода зависит от многих параметров, таких как количество хостов, на которые планируется осуществлять установку, пропускная способность сетевых каналов взаимодействия, конфигурация ОС и многого другого.

ОС Linux предоставляет пользователю следующие варианты установки.

- **Установка с компакт-диска.**
- **Установка с жесткого диска.** Требуется наличие образов формата ISO инсталляционных компакт-дисков на разделах жесткого диска, доступных для программы-установщика. Кроме того, необходим специальный *загрузочный диск*, созданный из файла образа **boot.iso**, расположенного на первом инсталляционном компакт-диске.
- **Сетевая установка** (через сервисы NFS, FTP или HTTP).
- **Автоматизированная установка (kickstart).** Установка осуществляется в автоматизированном режиме в соответствии с заранее подготовленным файлом формата **kickstart**. Метод **kickstart** может быть использован как при установке с компакт-диска, по сети или при помощи технологии PXE.
- **Установка при помощи технологии PXE.**
- **Установка с помощью сервиса RHN**¹² (Red Hat Network Provisioning). Установка данного типа предполагает наличие специальной подписки на сервисы **RHN Provisioning module** и **RHN Satellite Server**. Веб-интерфейс сервиса **RHN Satellite Server** имеет встроенные средства создания скриптов формата **kickstart**, при помощи которых можно настроить несколько инсталляционных профилей. Дальнейший процесс установки осуществляется удаленно в запланированное время.

10 Более подробно о сервисе RHN можно узнать по адресу https://www.redhat.com/f/pdf/rhn/rhn101698_0705.pdf

11 Под инсталляционным ресурсом понимается дистрибутив ОС Linux.

12 Подробную информацию о сети RHN можно получить из источника http://www.redhat.com/magazine/010aug05/features/rhn_overview/

1.4.2. Создание инсталляционного ресурса

Любая приобретенная подписка Red Hat Enterprise Linux дает возможность доступа к сети RHN, в которой содержатся последние дистрибутивы ОС Linux. Каждый дистрибутив ОС Linux состоит из нескольких файлов образов, упакованных в формате ISO. Данные файлы впоследствии могут использоваться для создания инсталляционного ресурса.

Сетевая установка ОС Linux, в том числе методом **kickstart**, при отсутствии первого инсталляционного компакт-диска, может быть начата при помощи специального загрузочного диска, сделанного из файла **boot.iso**, расположенного в каталоге **/images** первого инсталляционного образа. Если загрузка ведется при помощи внешнего USB-диска, вместо файла **boot.iso** необходимо использовать файл **diskboot.img**.

Для создания загрузочного диска необходимо выполнить следующие действия:

1. Создать каталог, куда будет осуществляться монтирование инсталляционного файла образа формата ISO, с именем **/tmp/inst**:

```
mkdir /tmp/inst/
```

2. Смонтировать первый инсталляционный образ.

```
mount -o loop <имя_первого_файла_образа>.iso /tmp/inst/
```

3. Записать файл образа **boot.iso** на диск:

```
cdrecord dev=/dev/hdc -v -eject /tmp/inst/images/boot.iso
```

или записать файл образа **diskboot.img** на USB-накопитель:

```
dd if=/tmp/inst/images/diskboot.img of=/dev/sda13
```

Для создания инсталляционного ресурса, использующего сервис NFS, необходимо выполнить следующие действия:

1. Создать каталог и поместить в него все инсталляционные файлы образов, полученные из сети **RHN**.
2. Настроить доступ к каталогу, содержащему все инсталляционные файлы образов, по NFS. Для этого необходимо добавить в файл **/etc/exports** следующую строку:

```
/inst *(ro, sync)
```

3. Добавить каталог **/inst** в список экспортируемых файловых систем:

```
exportfs -av
```

4. Активизировать сервис NFS:

```
service nfs restart
```

1. Для создания инсталляционного ресурса, использующего доступ к инсталляционным файлам по протоколу HTTP, необходимо выполнить следующие действия: Создать каталог, в котором будут располагаться инсталляционные файлы образов ОС Linux. Данный каталог должен

¹³ При выполнении данной команды все данные на внешнем USB диске будут удалены.

располагаться в корневом каталоге веб-сервера¹⁴. Смонтировать инсталляционные файлы образов и скопировать их содержимое в каталог инсталляционного ресурса.

2. Назначить пользователя **apache** и группу **apache** владельцами созданного каталога инсталляционного ресурса:

```
chown -R apache:apache /var/www/html/inst
```

3. Перезапустить сервис **httpd** и проверить доступность файлов по адресу **http://<имя_веб_сервера>/inst**.

Создание инсталляционного ресурса, использующего доступ к инсталляционным файлам по протоколу FTP, выполняется аналогично. Файлы помещаются в каталог **/var/ftp/pub/inst** и перезапускается сервис **vsftpd**. Доступность можно проверить с помощью ftp-клиента по адресу **ftp://<имя_FTP_сервера>/pub/inst**.

В случае выбора сетевых методов инсталляции рекомендуется использовать метод NFS, так как доступ к файлам осуществляется быстрее, чем при использовании остальных сетевых.

1.4.3. Процесс установки

Существуют следующие варианты начальной загрузки инсталляционного процесса:

- с инсталляционных CD- или DVD-дисков ОС Linux;
- с первого инсталляционного CD- или DVD-диска ОС Linux для сетевой установки;
- с загрузочного CD- или USB-диска ОС Linux;
- с сетевой карты с помощью технологии PXE.

После выбора в BIOS необходимого типа загрузочного устройства и начала загрузки с него, будет отображен основной инсталляционный экран ОС Linux. На этом экране в приглашении командной строки можно указать методы установки, параметры загрузки ядра ОС Linux и другие параметры инсталляции. Список доступных опций можно просмотреть, нажав на клавишу **<F2>**.

В случае инсталляции ОС Linux с CD/DVD компакт-дисков можно проверить внешний носитель на наличие ошибок, введя команду **linux mediacheck**.

Для начала процесса инсталляции необходимо ввести команду в приглашение **boot**. Возможны следующие варианты команд:

- **linux** (или простое нажатие **<Enter>**) – процесс будет произведен в графическом режиме;
- **linux text** – процесс будет произведен в текстовом режиме;
- **linux askmethod** – с помощью данной команды указывается место, где хранятся инсталляционные файлы ОС Linux; команда используется в ос-

¹⁴ В данном случае используется веб-сервер Apache, входящий в состав дистрибутива ОС Linux.

новном при сетевом типе инсталляции или инсталляции с жесткого диска. Этот вариант выбирается автоматически в случае загрузки с загрузочного компакт или USB-диска.

Весь процесс установки можно разделить на следующие этапы:

- 1) Выбор метода установки;
- 2) Выбор языка, который будет использован при установке;
- 3) Выбор основной раскладки клавиатуры, которая будет использоваться в системе;
- 4) Регистрация системы (ввод регистрационного ключа);
- 5) Разметка дискового пространства;
- 6) Настройка параметров загрузчика GRUB;
- 7) Настройка сетевых интерфейсов;
- 8) Выбор часового пояса;
- 9) Задание пароля пользователя root;
- 10) Выбор метода конфигурации пакетов (по умолчанию или вручную);
- 11) Выбор устанавливаемых групп пакетов (если был выбран ручной метод конфигурации пакетов);
- 12) Непосредственная установка выбранных групп пакетов;
- 13) Послеинсталляционные этапы, включая:
 - принятие лицензионного соглашения;
 - базовая настройка межсетевого экрана;
 - базовая настройка SELinux;
 - базовая настройка Kdump;
 - настройка даты и времени;
 - регистрация системы и настройка обновлений пакетов программ;
 - создание дополнительных учетных записей пользователей;
 - настройка звукового адаптера;
 - установка пакетов с дополнительных дисков Red Hat.

Особое внимание следует уделить разметке дискового пространства и выбору устанавливаемых пакетов. При инсталляции ОС Linux возможны следующие варианты разметки дискового пространства:

- удаление разделов Linux на выбранных жестких дисках и разметка дискового пространства по умолчанию (**Remove linux partitions on selected drives and create default layout**).

- удаление всех разделов на выбранных жестких дисках и разметка дискового пространства по умолчанию (**Remove all partitions on selected drives and create default layout**) (табл. 1.6.);
- использование всего доступного свободного пространства на выбранных жестких дисках и разметка дискового пространства по умолчанию (**Use free space on selected drives and create default layout**);
- разметка дискового пространства вручную (**Create custom layout**);
- Пункт «**Advanced storage configuration**» позволяет добавить к локальным жестким дискам дополнительные разделы с удаленных хостов или систем хранения данных (СХД) при помощи протокола **iSCSI**¹⁵;
- При разметке дискового пространства на этапе инсталляции ОС Linux существует возможность просмотра и редактирования созданной по умолчанию конфигурации разделов. Для этого необходимо отметить пункт «**Review and modify partitioning layout**».

Корневой раздел / является обязательным разделом ОС Linux. Для архитектур x86 и x86_64 также рекомендуется создавать разделы **swap** и **/boot**, используемые для подкачки и загрузки ОС Linux соответственно. По умолчанию создаются разделы **/boot**, **/swap** и **/**, причем последние два создаются на основе технологии управления логическими томами – **LVM**. В других архитектурах, например в Itanium, вместо раздела **/boot** рекомендуется создавать раздел **/boot/efi**, который должен быть первичным разделом. Администратор может создать дополнительные отдельные разделы – **/home** (для хранения пользовательских данных) или **/var/www** (для хранения веб-контента). Разделы могут быть созданы на основе технологии **LVM** или **RAID**¹⁶ и содержать свою собственную файловую систему.

Таблица 1.6. Разметка дискового пространства по умолчанию

Раздел	Объем дискового пространства
/boot	100Мб
swap	Удвоенное количество доступной оперативной памяти
/	Оставшееся свободное пространство, созданное на основе LVM

В окне редактирования конфигурации дисковой подсистемы для каждого созданного раздела отображается следующая информация:

- **Физическое устройство (Device)**. Если используются жесткие диски с интерфейсом IDE, этот пункт имеет вид **/dev/hdXY**, где символу **Y** соот-

¹⁵ В ОС Linux программная реализация взаимодействия по протоколу iSCSI представлена в проекте Linux-iSCSI, ресурсы которого находятся по адресу <http://linux-iscsi.sourceforge.net/>.

¹⁶ Избыточный массив независимых дисков **RAID** (Redundant Array of Independent Disks) – это система организации дисков, в которой данные распределяются или дублируются на несколько жестких дисках.

ветствует порядковый номер раздела, а символу **X** соответствуют буквы:

- **a** – master диск, расположенный на первом контроллере IDE;
- **b** – slave диск, расположенный на первом контроллере IDE;
- **c** – master диск, расположенный на втором контроллере IDE;
- **d** – slave диск, расположенный на втором контроллере IDE.

В случае использования жестких дисков с интерфейсом SCSI, физическое устройство каждого раздела представляется записью `/dev/sdXY`, где символу **Y** соответствует порядковый номер раздела, а символу **X** соответствуют буквы:

- **a** – диск с идентификатором (SCSI ID), равным 1;
- **b** – диск с идентификатором (SCSI ID), равным 2;
- и т.д.
- **Точка монтирования, устройство RAID или группа томов LVM (Mount Point/RAID/Volume)**. Точка монтирования представляет собой раздел, к которому монтируется физическое устройство, например `/mnt`. Об остальных вариантах представления (устройство RAID или группа томов LVM) будет рассказано далее.
- **Тип файловой системы (Type)**. Данное поле отображает тип файловой системы раздела, например, **ext3, software RAID**.
- **Формат (Format)**. Данное поле содержит отметки для разделов, которые необходимо отформатировать.
- **Размер (Size (MB))**.
- **Начало (Start)**. Данное поле содержит порядковый номер цилиндра, с которого начинается раздел.
- **Конец (End)**. Данное поле содержит порядковый номер цилиндра, которым оканчивается раздел.

ОС Linux является достаточно гибкой ОС с точки зрения требования к размеру разделов. В зависимости от количества устанавливаемых программ и поддерживаемых языков, суммарный размер установки ОС Linux может занимать от 2 до 10 Гб и более. Необходимость создания данных разделов обусловлена установленными в ОС Linux сервисами – электронной почтой, веб-сервисом или сервис базы данных и др., для хранения данных которых рекомендуется создавать отдельные разделы. Размеры данных разделов рассчитываются для каждого сервиса отдельно.

Таблица 1.7. Общие рекомендации по размерам разделов ОС Linux.

Раздел	Размер раздела	Рекомендации по расчету размера раздела
swap	от 256 Мб	Если в системе присутствует до 2 Гб оперативной памяти, то размер должен быть вдвое больше, размера оперативной памяти. Если присутствует более 2 Гб оперативной памяти, то размер раздела должен быть вдвое больше, чем размер доступной оперативной памяти плюс 2 Гб. В случае использования количества оперативной памяти более 32 Гб, размер раздела swap может быть равен размеру оперативной памяти или быть меньше размера оперативной памяти. Выбор размера раздела swap также зависит от требований конкретных приложений, которые планируется устанавливать.
/boot	100 Мб	В данном разделе содержится ядро ОС Linux и программная часть загрузчика GRUB. Как правило, размер данного раздела может быть меньше 100 Мб.
/	3 - 5 Гб	При минимальной установке, размер корневого раздела должен быть более 3 Гб. Обычно размер корневого раздела зависит от того, созданы ли остальные разделы (/home , /usr и т.д.), как отдельные разделы.
/home	любой	В данном разделе как правило размещаются данные пользователей. Если компьютер предполагается использовать как сервер, не содержащий данных обычных пользователей, раздел можно не использовать.
/usr	5 - 10 Гб	Если на компьютер устанавливается поддержка дополнительных системных языков ОС, для данного раздела может потребоваться более 10 Гб свободного пространства.
/tmp	≥ 1 Гб	В данном разделе содержатся временные файлы приложений.
/var	≥ 5 Гб	В данном разделе хранятся главным образом файлы журналов различных сервисов ОС Linux и кэшированные данные программ установки ПО. В ОС Linux можно настроить ротацию файлов журнальных в целях экономии дискового пространства.
/opt	любой	Данный раздел в основном используется коммерческим ПО, не входящим в состав дистрибутива ОС Linux (например, СУБД Oracle). Размер данного раздела необходимо выбирать, исходя из требований ПО, которое планируется устанавливать.

Настройка загрузчика ОС Linux позволяет указать подсистеме BIOS компьютера расположение ядра ОС Linux и дополнительных программ, необходимых для загрузки ОС.

Настройка часового пояса является достаточно важной задачей, поскольку рассогласование времени между серверами внутри сети и несоответствие времени на сервере действительному времени могут привести к неправильной работе определенных сервисов. В частности, если часовой пояс указан неправильно, дата и время получения письма может быть неверной.

Затем необходимо выбрать группы пакетов программ, которые необходимо установить. Некоторые группы пакетов, например **base**, устанавливаются по умолчанию в обязательном порядке. Количество доступных для установки групп пакетов зависит от приобретенной подписки Red Hat (Subscription), а так же типа дистрибутива ОС (Red Hat Enterprise Linux 5 Desktop или Red Hat Enterprise Linux 5 Server).

Базовую настройку групп пакетов желательно осуществить в процессе инсталляции. После инсталляции это также возможно сделать при помощи консольной программы **yum** или графической программы **pirut**. Полный список всех пакетов, входящих в состав групп пакетов, содержится в файле **comps-rhel5-server-core.xml**, расположенном в каталоге **/Server/repodata** первого инсталляционного образа ОС **Red Hat Enterprise Linux 5 Server**. В случае установки ОС **Red Hat Enterprise Linux 5 Desktop** список всех пакетов, входящих в состав групп пакетов, содержится в файле **comps-rhel5-client-core.xml**, расположенном в каталоге **/Client/repodata** первого образа. Все группы пакетов ОС Linux объединены в категории, представленные в Приложении 3.1.

Содержимое каждой группы пакетов можно просмотреть и при желании изменить, нажав на кнопку **Optional Packages** и отметив нужные для установки пакеты.

На этом процесс инсталляции не заканчивается. При первой загрузке ОС Linux выполняется программа **Red Hat Setup Agent**, в которой необходимо ответить на несколько вопросов, прежде чем ОС будет полностью готова к работе. Типовые ответы на данные вопросы приведены в практической работе к данному модулю.

1.4.4. Автоматизированная установка kickstart

Вопросы, обычно задаваемые при интерактивной инсталляции, можно записать в единый конфигурационный текстовый файл, пригодный для использования для автоматизированной установки ОС Linux. Данный метод используется в основном при установке по сети, поскольку если инсталляция производится с компакт-дисков, всё равно придется менять инсталляционные диски.

Общая последовательность действий по инсталляции ОС Linux с использованием метода **kickstart** следующая:

1. Создать инсталляционный ресурс и сделать его доступным для систем, на которые планируется установить ОС Linux.

2. Создать конфигурационный файл `kickstart`.
3. Создать загрузочный диск, если для загрузки инсталляции не используется технология PXE.
4. Скопировать конфигурационный файл `kickstart` в загрузочный диск или сделать его доступным через сетевой ресурс.
5. Запустить инсталляцию.

Конфигурационный файл **kickstart** представляет собой обычный текстовый файл, содержащий в каждой строке специальные директивы¹⁷, которые упорядочиваются в следующие секции:

- **Секция команд.** Содержит директивы для ответов на вопросы инсталляционной программы **anaconda**. Данные директивы могут располагаться в произвольном порядке, но обязательно должны быть определены в начале файла `kickstart`. Если указаны не все обязательные директивы, то этапы инсталляции, связанные с пропущенными обязательными директивами, необходимо будет выполнить в интерактивном режиме. Некоторые директивы, например, **install**, требуют указания дополнительных директив в отдельной строке.
- **Секция %package.** Данная секция позволяет устанавливать группы пакетов. Каждая группа пакетов указывается на отдельной строке, вначале которой ставятся символы **@** и пробел. Если названию пакета предшествует знак «-», то данный пакет устанавливаться не будет. Список групп пакетов для ОС RHEL 5 Server можно посмотреть в `/Server/repodata/comps-rhel5-server-core.xml`. Значения элемента `<id>` из данного файла, могут использоваться в файле `kickstart` для описания устанавливаемых групп пакетов. В элементах `<packagereq>` указываются пакеты, входящие в состав данной группы. Если в атрибуте **type** указано **default (optional)**, то пакеты устанавливаются (не устанавливаются) по умолчанию при выборе данной группы. Для установки опциональных пакетов в файле `kickstart` необходимо указать их названия. Группы пакетов **Core** и **Base** устанавливаются в любом случае без явного указания в секции `%package`. Если в секции `%package` указан параметр **--ignoremissing**, то в случае отсутствия указанной группы пакетов или отдельного пакета, установка будет продолжена без всякого оповещения пользователя.
- **Секция %pre.** В данной секции указываются скрипты или отдельные команды, которые необходимо запустить до начала процесса инсталляции. Данные команды выполняются сразу же после считывания файла `kickstart`. На данном этапе возможна работа с сетью, однако только на уровне IP-адресов, при обращении по DNS имени команда завершится с ошибкой. Сек-

¹⁷ С полным списком и описанием всех директив можно ознакомиться по адресу http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/System_Administration_Guide_/Kickstart_Installations-Kickstart_Options.html. Обязательные директивы помечены словом `required` в круглых скобках.

ция **%pre** должна следовать сразу за секцией **%package**.

- **Секция %post.** В данной секции указываются скрипты или отдельные команды, которые необходимо запустить после завершения процесса инсталляции. Секция **%post** должна быть описана после всех других секций файла **kickstart**. В данной секции обычно указываются команды установки дополнительных пакетов и настройки конфигурационных файлов. По умолчанию, запуск команд в секции **%post** осуществляется в среде с измененным корневым каталогом (**chroot**). В качестве корневого каталога используется каталог **/mnt/sysimage**. Для выполнения команды относительно реального корневого каталога в данную секцию необходимо добавить параметр **--nochroot**. Опция **--interpreter <интерпретатор>** позволяет указать командный интерпретатор, который будет использоваться при обработке скриптов, содержащихся в данной секции.

Существует два способа создания файла **kickstart**:

- Редактирование существующего файла **kickstart (/root/anaconda-ks.cfg)**, который был создан программой **anaconda** в процессе инсталляции ОС Linux. Этот файл можно в любом текстовом редакторе, однако следует обращать внимание на перенос строк, чтобы не допустить перенос части команды на следующую строку.
- Создание нового файла **kickstart** при помощи графической программы **Kickstart Configurator**, для запуска которой используется команда **system-config-kickstart**.

Наиболее удобным способом создания файла **kickstart** является использование графической программы **Kickstart Configurator**, которая содержит в себе все необходимые средства для создания файла **kickstart**.

После создания конфигурационного файла **kickstart** его необходимо сделать доступным для инсталляционной программы **anaconda**. Существуют следующие варианты хранения файла **kickstart**:

- в одном из разделов жесткого диска;
- на компакт-диске;
- HTTP, FTP или NFS ресурс;
- на загрузочном диске.

Рассмотрим перенос файла **kickstart** на загрузочный диск. Файл **kickstart** должен иметь название **ks.cfg** и должен располагаться в корневом каталоге загрузочного диска. Для создания загрузочного диска, содержащего файл **kickstart**, необходимо выполнить следующие действия:

1. Смонтировать образ загрузочного диска и скопировать все его содержимое в каталог **/tmp/bootks**.
2. В каталоге **/tmp/bootks** установить права на запись для каталога **isolinux** и скопировать туда файл **kickstart**.

3. Создать образ загрузочного диска, содержащего файл kickstart:

```
mkisofs -o bootks.iso -b isolinux.bin -c boot.cat -no-emul-boot \
  -boot-load-size 4 -boot-info-table -R -J -v -T isolinux/
```

4. Записать образ загрузочного диска на внешний носитель

Если файл kickstart размещен на ресурсе, доступном по NFS, можно автоматизировать выбор файла kickstart при помощи DHCP сервера. Для этого необходимо:

1. На NFS-сервере, содержащем инсталляционный ресурс, создать каталог - например, с именем **/kickstart**
2. Добавить в конфигурационный файл DHCP-сервера (**dhcpd.conf**) следующие строки, указывающие на расположение файла kickstart:

```
filename "/kickstart/";
next-server my.nfs.server;
```

где **my.nfs.server** – DNS имя сервера NFS. Если этот параметр не указывать, программа инсталляции будет считать, что сервисы NFS и DHCP располагаются на одном хосте.

3. Создать в каталоге **/kickstart** файлы kickstart с названием **<IP-адрес>-kickstart**, где **<IP-адрес>** – реальный IP-адрес хоста, на который производится установка kickstart. Данный IP-адрес присваивается хосту автоматически по средствам DHCP.

После того как процесс установки запущен при помощи созданного загрузочного диска или любым другим методом, например, через NFS, в приглашении **boot:** необходимо ввести специальную команду, которая зависит того, где располагается файл kickstart:

- CD/DVD компакт-диск – **linux ks=cdrom:/ks.cfg;**
- Ресурс NFS – **linux ks=nf:<сервер>:/<имя_файла>**, если в конце имени файла указать символ /, то программа инсталляции распознает данный файл как каталог, в котором будет осуществлен поиск файла kickstart с именем **<IP-адрес>-kickstart**. Вместо фрагмента **<IP-адрес>** будет подставлен реальный IP-адрес, полученный хостом по DHCP;
- Ресурс HTTP – **linux ks=http://<сервер>/<имя_файла>**;
- Гибкий диск – **linux ks=floppy:/<имя_файла>**;
Файловая система гибкого диска, содержащая файл kickstart, должна быть типа **ext2** или **vfat**.
- Жесткий диск (ext2 или vfat) – **linux ks=hd:<устройство>:/<имя_файла>**;
Вместо фрагмента **<устройство>** необходимо указать устройство, которым представлен раздел, содержащий файл kickstart (например, **sda1**);
- Ресурс NFS описанный в настройках сервиса DHCP – **linux ks**.

1.4.5. Установка с использованием технологии PXE

Большинство современных сетевых адаптеров поддерживают загрузку, используя механизм **Pre-Execution Environment (PXE)**. После того, как BIOS компьютера определит, что загрузка системы должна осуществляться через сетевой адаптер, происходит выполнение клиентской программы, которая посылает широковещательные запросы на DHCP-сервер. Если в сети имеется DHCP-сервер, настроенный на предоставление информации о сервере TFTP и расположении инсталляционных файлов ОС Linux, клиент PXE запускает сетевую инсталляцию ОС Linux. Более того, некоторые сетевые адаптеры поддерживают функцию «Wake-up on LAN», при помощи которой можно включить компьютер посылая на сетевой адаптер специальный пакет **magic**.

Использование механизма PXE¹⁸ может применяться совместно с инсталляцией kickstart. Данная комбинация позволяет администратору загружать удаленное оборудование и запускать на нем сетевую инсталляцию ОС Linux в автоматизированном режиме kickstart.

1.5. Управление пакетами программ

После завершения инсталляции ОС Linux прежде чем система будет полностью готова к работе, может понадобиться установить дополнительные пакеты программ, обновить текущие версии пакетов или же настроить сервис обновления пакетов **Red Hat Network (RHN)**.

Традиционно пакеты использовались для распространения программного обеспечения, однако их можно применять и для распространения конфигурационных файлов. Пакеты имеют ряд следующих преимуществ перед обычными архивами, такими как **tar.gz**:

- простота обновления;
- возможность запроса информации о пакетах в базе RPM;
- проверка корректности файлов пакетов перед их установкой;
- установка программ из оригинального исходного кода, с применением заданных наборов исправлений и инструкций компилятора.

Важно отметить, что процесс инсталляции пакета протекает как одна транзакция. Единственным типом пакетов, используемых в ОС Linux, является **rpm**. Для того чтобы использовать программное обеспечение, которое предоставляет пакет **rpm**, его необходимо, прежде всего установить в систему. Для установки пакета **rpm** в систему используется специальное программное обеспечение, называемое **менеджером пакетов**. С помощью менеджеров пакетов в общем случае выполняют следующие задачи:

¹⁸ Подробно о сетевой установке ОС Linux с использованием технологии PXE можно узнать здесь: https://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/System_Administration_Guide/_PXE_Network_Installations.html

- установка новых пакетов;
- обновление установленных пакетов;
- проверка целостности файлов, входящих в состав пакета;
- запрос установленных пакетов и их параметров;
- удаление установленных пакетов;
- создание пакетов из исходных кодов.

1.5.1. Менеджер пакетов RPM

В общем случае пакет **rpm** представляет собой контейнер, содержащий группы файлов, ассоциированных с определенным приложением. В состав групп входят бинарные исполняемые файлы, библиотеки функций, конфигурационные файлы и файлы документации. Кроме того, в состав каждого **rpm**-пакета входит набор инструкций, которые определяют, как и куда следует установить файлы пакета, а также позволяющие определить необходимые зависимости файлов данного пакета от файлов, содержащихся в других пакетах.

Каждый **rpm**-пакет имеет строго определенное название, в состав которого входит версия пакета, релиз пакета и архитектура процессора, с которой данный пакет можно использовать. Структура именованного rpm пакета имеет следующий вид:

`<имя пакета>-<версия пакета>-<релиз>-<архитектура>.rpm`

Версия пакета диктуется непосредственным разработчиком ПО. Новые версии включают новые возможности. Релиз, как правило, относится к одной версии и содержит исправления ошибок в данной версии или незначительные усовершенствования. Архитектура определяет, для какого типа процессора предназначен данный пакет. Существуют пакеты, независимые от процессорной архитектуры и устанавливаемые на оборудовании с любыми процессорами.

Наиболее используемые архитектуры, использующиеся в пакетах **rpm**, перечислены в табл. 1.8.

Таблица 1.8. Наименования пакетов RPM

Архитектура	Описание
noarch	Архитектурно-независимые пакеты, которые могут быть установлены на оборудовании с произвольной архитектурой процессора
i386	Предназначены для установки на 32-х битных системах типа x86
i586	Предназначены для установки на устаревших системах x86
i686	Предназначены для установки в системах, имеющих процессоры типа Intel Pentium I, II, III, IV
x86_64	Предназначены для установки на 64-х битных системах с процессорами типа AMD Opteron, AMD Athlon64, Intel EM64T

Архитектура	Описание
ia64	Данные пакеты предназначены для установки на системы с процессорами типа Intel Itanium
ppc	Данные пакеты предназначены для установки на 32-х битные системы IBM, такие как IBM eServer, IBM pSeries, IBM iSeries, IBM Power
s390x	Данные пакеты предназначены для установки на 64-х битные системы IBM eServer System z

Пакеты **rpm** могут находиться на локальных файловых системах, но могут быть доступны и по сети. Основным средством управления пакетами в ОС Linux является менеджер пакетов **Red Hat Package Manager (RPM)**. В состав данного менеджера пакетов входит утилита **rpm**, при помощи которой происходит установка, обновление и выполнение других операций над пакетами, а также специальная *база данных*, в которой содержится информация обо всех установленных в системе пакетах, включая зависимости между пакетами и принадлежащие пакетам файлы. Кроме того, менеджер пакетов RPM отслеживает целостность файлов, входящих в состав пакета и управляет конфликтами, происходящими в процессе установки или обновления пакетов.

Перед установкой пакета **rpm** желательно убедиться, что он предоставлен из достоверного источника. Процесс проверки подлинности пакета заключается в проверке цифровой подписи открытым ключом, предоставленным разработчиком пакета. Подписка пакетов требует использования специальных утилит, использующих стандарт **OpenPGP (RFC4880)**, использующих набор ключей для шифрования и дешифрования данных. Одной из таких утилит является **GnuPG**, которая используется для подписи RPM пакетов. Используя **GnuPG**, разработчик создает открытый и закрытый ключ. Закрытый ключ остается в надежном месте у разработчика, а открытый ключ публикуется в общедоступном месте, доступном по протоколам HTTP или FTP.

В процессе установки или обновления **rpm** пакетов проверка цифровой подписи выполняется автоматически менеджером пакетов RPM.

Прежде чем использовать возможность проверки цифровых подписей пакетов, необходимо импортировать в систему соответствующие открытые ключи разработчиков данных пакетов. Импорт ключей для проверки цифровых подписей осуществляется командой `rpm --import <файл>`, указав файл, содержащий ключи в формате GPG. Файлы GPG, распространяемые компанией Red Hat, располагаются в корневом каталоге первого инсталляционного компакт-диска ОС Linux и начинаются со строки **RPM-GPG-KEY**. Для проверки количества импортированных открытых (public) ключей используется команда `rpm -qa gpg-pubkey`. Для просмотра детальной информации по импортированным ключам необходимо использовать команду `rpm -qi <название_ключа>`.

После того, как необходимые открытые ключи будут импортированы в систему, убедиться в том, что пакет был получен из достоверного источника можно при помощи команды `rpm -K <название_пакета.rpm>`. Если пакет не подписан,

вывод данной команды будет иметь вид «**NOT OK**». Если необходимый ключ не был импортирован в систему, то будет выдано сообщение вида «**MISSING KEY**». Если пакет подписан и не был изменен после того, как его подписал удостоверенный источник, будет выдано сообщение, содержащее фразу «**md5 gpg OK**».

Для установки пакетов **rpm** необходимо запустить команду **rpm** следующим образом:

```
rpm -ivh <название_пакета1.rpm> <название_пакета2.rpm>  
<название_пакетаN.rpm>
```

Ключи **-ivh** используются для установки пакета, детального отображения процесса установки и отображения прогресса установки. При указании нескольких названий пакетов будут установлены все указанные пакеты. Запуск команды **rpm** с ключом **-i** осуществит установку пакета только в том случае, если в системе данный пакет еще не установлен. Для обновления уже установленного пакета используется ключ **-U** (если обновляемый пакет еще не установлен, то будет выполнена установка пакета). При использовании команды **rpm** с ключом **-F** установленный пакет будет обновлен, но если он отсутствует, то установка проведена не будет.

Некоторые файлы, входящие в состав пакета **rpm** (например, конфигурационные), помечаются специальным образом с тем, чтобы при обновлении пакета у пользователя была возможность сохранить прежние версии.

Для удаления пакета используется команда **rpm** с ключом **-e**. Если пакет, который вы пытаетесь удалить, необходим для работы других пакетов, система выдаст предупреждение о нарушении зависимостей в случае удаления данного пакета. Если пакет, который необходимо удалить, содержит конфигурационные файлы, которые были изменены, то менеджер пакетов сохранит их с расширением **.rpmsave**.

Помимо процедур установки, обновления и удаления пакетов менеджер RPM позволяет выполнять проверку целостности файлов, входящих в состав пакета, согласно таким параметрам как код доступа, владелец файла, размер файла, MD5 сумма. Проверка данного типа заключается в сличении параметров оригинального файла с параметрами файла, установленного в системе. Для выполнения данной проверки необходимо запустить **rpm -V <название_пакета>**. Для проверки всех пакетов в системем можно использовать команду **rpm -Va**, в результате выполнения которой отобразятся все файлы, которые были изменены.

Если вывод данной команды окажется пустым, значит файлы, входящие в состав пакеты не были изменены. В случае, если какие-либо файлы были изменены, то отобразится таблица, содержащая список измененных файлов и соответствующий код, обозначающий измененный параметр. В следующем листинге приведен пример проверки файлов, входящих в состав пакета **sysklogd-1.4.1-40.el5**:

```
# rpm -Vv sysklogd-1.4.1-40.el5  
..... c /etc/logrotate.d/syslog  
..... c /etc/rc.d/init.d/syslog  
..... c /etc/sysconfig/syslog  
S.5....T c /etc/syslog.conf  
..... /sbin/klogd  
..... /sbin/syslogd
```

```

..... /usr/share/doc/syslogd-1.4.1
..... d /usr/share/doc/syslogd-1.4.1/ANNOUNCE
..... d /usr/share/doc/syslogd-1.4.1/CHANGES
..... d /usr/share/doc/syslogd-1.4.1/ChangeLog

```

Как видно из листинга, файл `/etc/syslog.conf` был изменен, а именно:

- был изменен размер файла, о чем свидетельствует символ **S** в поле кода проверки;
- сумма MD5 была изменена, о чем свидетельствует символ **5** в поле кода проверки;
- время модификации файла, о чем свидетельствует символ **T** в поле кода проверки.

Следом за кодом проверки указывается тип файла. Возможные коды проверки файлов, входящих в состав пакета **rpm**, приведены в табл. 1.9.

Таблица 1.9. Коды проверки RPM

Код проверки	Описание
S	Размер файла был изменен
M	Код доступа файла был изменен
5	Сумма MD5 файла была изменена
D	Старший или младший номер устройства был изменен
L	Путь символической ссылки был изменен
U	Владелец файла был изменен
G	Группа файла была изменена
T	Время последнего изменения файла было изменено

База данных RPM, входящая в состав менеджера пакетов RPM, содержит много полезной информации, которую можно получить, используя специальные запросы. Данные запросы позволяют определить, какому пакету принадлежит присутствующий в системе файл, определить файлы, входящие в состав пакета, а также просмотреть дополнительную информацию о пакете. Для запроса к базе RPM необходимо выполнить команду **rpm** с ключом **-q<N>**, где **<N>** – опция, зависящая от типа запроса (таблице 1.10).

Таблица 1.10. Опции команды **rpm -q**

Команда	Описание
<code>rpm -qa</code>	Отобразить все установленные в системе пакеты
<code>rpm -qf <имя_файла></code>	Определить пакет, к которому принадлежит указанный файл
<code>rpm -qc <название_пакета></code>	Отобразить только конфигурационные файлы, входящие в состав указанного пакета

Команда	Описание
<code>rpm -qi <название_пакета></code>	Отобразить краткую информацию о пакете
<code>rpm -ql <название_пакета></code>	Отобразить все файлы, входящие в состав пакета
<code>rpm -qR <название_пакета></code>	Отобразить все зависимости, без которых пакет нельзя установить
<code>rpm -qd <название_пакета></code>	Отобразить файлы документации, входящие в состав пакета
<code>rpm -q --changelog <название_пакета></code>	Отобразить список изменений, отображающий изменения между различными версиями данного пакета

Помимо стандартных запросов обращения, база данных RPM позволяет использовать расширенный формат запросов на основе специальных тэгов. Для этого используется ключ **--qf (--queryformat)**, в котором задается формат запроса подобный функции **printf** языка C:

```
rpm -qf %{tag_name}
```

Здесь в качестве параметра **tag_name** указывается один или несколько тегов¹⁹, информацию о которых можно получить, выполнив команду **rpm --querytags**.

Например, для отображения информации по лицензиям всех установленных пакетов можно использовать следующий запрос:

```
# rpm -qa --qf "[%-50{NAME} %{LICENSE}\n]"
firefox                               MPLv1.1 or GPLv2+ or
LGPLv2+
ekiga                                  GPL
gstreamer-plugins-good                LGPL
gnome-python2-applet                  GPL/LGPL
gnome-session                          GPL
xorg-x11-drv-dmc                       MIT/X11
```

Если необходимо получить информацию о времени установки пакетов, то можно использовать следующий запрос:

```
# rpm -qa --qf "%{NAME}-%{VERSION}-%{RELEASE} %{INSTALLTIME:date}\n"
bind-libs-9.3.4-6.P1.e15 Tue 18 Nov 2008 07:33:42 PM MSK
apr-util-1.2.7-7.e15 Tue 18 Nov 2008 07:33:45 PM MSK
nscd-2.5-24 Tue 18 Nov 2008 07:33:57 PM MSK
...
```

Непосредственно файлы базы данных RPM располагаются в каталоге **/var/lib/rpm** и представляют собой файлы в формате BDB²⁰ (Berkeley DB). В этом каталоге содержатся файлы следующих типов:

¹⁹ Описание доступных тэгов можно посмотреть здесь: <http://www.rpm.org/max-rpm/ch-queryformat-tags.html>

²⁰ Формат BDB используется для хранения информации подобно реляционным базам данных, но данные хранятся в более простом виде, обеспечивающим высокопроизводительный доступ (http://en.wikipedia.org/wiki/Berkeley_DB).

- **_db.001**, **_db.002** и т.д. – Файлы блокировок, используемые менеджером RPM.
- **Packages** – Содержит информацию о заголовках всех установленных пакетов в индексированном виде.
- **Name**, **Providename**, **Group** – используются для ускоренного доступа к соответствующей информации (название пакета, группа и т.д.).

В целях безопасности и надежности системы, данный каталог необходимо регулярно резервировать. В случае повреждения базы данных RPM можно выполнить ее пересоздание на основе данных об установленных пакетах, используя команду **rpm --rebuilddb.**, однако для этого обязательно наличие хотя бы одного файла – **/var/lib/rpm/Packages**. Существует так же возможность создать новую базу данных RPM, используя команду **rpm -initdb**, однако практической пользы в этом нет, так как она не будет содержать никаких данных об установленных пакетах. В случае невозможности восстановить базу данных RPM рекомендуется полностью переустановить систему.

В ОС Linux существуют графические программы управления пакетами **rpm**, такие как **pup** и **pirut**.

Если вы успешно зарегистрировались в сети RHN, то для обновления пакетов ОС Linux можно использовать программу **pup**, которую можно запустить из меню «**Applications**» → «**System Tools**» → «**Software Updater**». Данная программа отображает только те пакеты, для которых доступны обновления в сети RHN.

Для управления пакетами **rpm** в среде GNOME используется графическая программа **pirut**, которая позволяет устанавливать сразу несколько пакетов с разрешением соответствующих им зависимостей и отображать список уже установленных пакетов. Кроме того, с помощью данной программы можно искать пакеты по базе данных RPM. Запуск программы **pirut** осуществляется из меню «**Applications**» → «**Add/Remove Software**».

1.5.2. Менеджер пакетов YUM

Основным недостатком менеджера пакетов RPM является то, что он не разрешает зависимости пакетов в момент их установки. Каждый зависимый пакет необходимо указывать в командной строке, что может быть достаточно трудоемкой задачей. Для решения подобных проблем в ОС Linux имеется менеджер пакетов **YUM (Yellowdog Updater, Modified)**. Данный менеджер пакетов использует централизованное место хранения пакетов – репозиторий. Репозитории могут находиться на локальных файловых системах, HTTP и FTP серверах.

Прежде чем его использовать менеджер пакетов **YUM**, необходимо настроить подключение к сети RHN или же прописать дополнительные репозитории, содержащие пакеты RPM.

Менеджер пакетов **YUM** обладает следующими достоинствами:

- автоматическое разрешение зависимостей пакетов;
- существуют консольная и графическая версии программы;

- можно использовать несколько репозиториев, содержащих пакеты RPM;
- имеется возможность устанавливать пакеты только определенной версии или архитектуры.

Общий синтаксис консольной утилиты управления менеджером **YUM** следующий:

```
yum <команда> <название_пакета1> <название_пакета2>
```

Для установки пакета при помощи менеджера пакетов **YUM** используется следующий синтаксис команды:

```
yum install <название_пакета1> <название_пакета2> ...
```

Для установки конкретной версии пакета необходимо запустить команду в следующем виде:

```
yum install <название_пакета>--<версия>
```

Для установки пакета для определенной архитектуры нужно использовать ключи, как показано ниже:

```
yum install <название_пакета>.<архитектура>
```

Для обновления пакета необходимо запустить:

```
yum update <название_пакета>
```

Если при указании команды **yum update** будет опущено название пакета, то будет выполнена попытка обновить все установленные в системе пакеты.

Для удаления пакета необходимо запустить команду со следующими ключами:

```
yum remove <название_пакета>
```

При удалении указанного пакета также удаляются и зависимые пакеты. Напомним, менеджер пакетов RPM этой возможностью не обладает.

Для поиска пакетов в репозиториях используется следующая команда:

```
yum search <название_пакета>
```

Для выполнения различных действий связанных с управлением пакетами rpm, программа **yum** использует конфигурационный файл **/etc/yum.conf**, в котором указаны параметры репозиториев, содержащих пакеты RPM. В частности, в данном файле указываются параметры кэширования данных о пакетах, проверки цифровых подписей GPG, URI адреса репозиториев. Для просмотра всех директив конфигурационного файла программы **yum** можно выполнить команду **man yum.conf**. Как правило, придерживаются следующей конфигурации менеджера **YUM**: в конфигурационном файле **/etc/yum.conf** указываются глобальные настройки менеджера, а в каталоге **/etc/yum.repos.d** создаются файлы с описанием репозиториев и их параметров. Типовой файл для определения произвольного репозитория **YUM** имеет следующий вид:

```
[название_репозитория]
# комментарий
атрибут=значение
:
атрибут=значение
```

Здесь после указания формального названия репозитория в квадратных скобках перечисляются его основные параметры. В следующем листинге приведен пример локального репозитория, содержащего пакеты ОС Linux:

```
[Server]
gpgenabled=1
gpgkey=file:///var/www/html/yum/RPM-GPG-KEY-redhat-release file:///var/www/html/
yum/RPM-GPG-KEY-redhat-beta
name=Server
baseurl=file:///var/www/html/yum/Server/
enabled=1
```

В данном листинге первые два атрибута указывают на то, что необходимо выполнять проверку цифровых подписей пакетов перед их установкой. Атрибут **name** задает краткое описание репозитория. Атрибут **baseurl**, записанный в формате **протокол://расположение**, задает расположение файлов репозитория. Последний атрибут **enable** свидетельствует об активации использования данного репозитория.

В целях экономии трафика и ресурсов сети рекомендуется иметь собственный локальный репозиторий пакетов RPM, с периодически выполняемой синхронизацией пакетов с внешним репозиторием пакетов RPM. Для создания локального репозитория и его последующей синхронизации используются команды **createrepo** (пакет createrepo) и **reposync** (пакет yum-utils) соответственно. Общая последовательность действий для создания локальных репозиториях подробно описана здесь <http://yum.baseurl.org/wiki/RepoCreate>.

1.5.3. Создание пакетов RPM

Умение создавать пакеты RPM по большей части необходимо разработчику, но также будет полезно системному администратору при решении следующих задач:

- Распространение сценариев, конфигурационных файлов и шаблонов на несколько систем;
- Установка программ с измененными параметрами компиляции;
- Упаковка программ в пакеты RPM;

Использование для этих целей пакетов RPM дает следующие преимущества:

- Контроль версий. Поскольку каждый RPM пакет содержит информацию о версии, то отслеживание изменений в процессе длительного функционирования ОС существенно упрощается.
- Удобство распространения. Установка RPM пакетов не требует ручной компиляции и может быть выполнена неквалифицированным пользователем, как на локальные, так и на удаленные системы.
- Контроль целостности. Используя опцию проверки RPM пакетов, администратор может легко отслеживать любые изменения в установленных файлах пакетов.

Создание пакетов RPM может выполняться из под пользователя **root** в спе-

циально отведенном месте или из под обычного пользователя в его домашнем каталоге. Последний вариант более предпочтителен с точки зрения безопасности. Общий процесс создания RPM пакета состоит из следующих этапов:

1. Подготовка окружения для сборки пакетов.
2. Создание файла спецификации (SPEC).
3. Подготовка и размещение файлов исходного кода или собственных файлов в сборочном окружении.
4. Создание пакета.
5. Подписка пакета (опционально).

Первым этапом в создании RPM пакета является подготовка окружения. Для этого в домашнем каталоге пользователя необходимо создать следующие каталоги:

- BUILD – в данном каталоге выполняется непосредственная компиляция исходного кода.
- RPMS – в данный каталог помещаются созданные бинарные пакеты RPM.
- SOURCES – данный каталог содержит файлы с исходным кодом.
- SPECS – данный каталог содержит файлы спецификаций.
- SRPMS – данный каталог содержит созданные RPM пакеты с исходным кодом.

Для создания данного окружения в домашнем каталоге пользователя создается файл **.rpmmacros** следующего содержания:

```
%_topdir /home/<username>/RPMBUILD
```

Данный файл указывает программе создания пакетов **rpmbuild** откуда брать все необходимые данные для сборки пакета. Кроме того, в данном файле могут быть указаны дополнительные параметры, такие как расположение файлов исходного кода и прочее. За образец для данного файла можно взять файл **/usr/lib/rpm/macros**, содержащий глобальные настройки сборки пакетов.

После создания файла **.rpmmacros** необходимо создать упомянутые каталоги:

```
mkdir RPMBUILD
mkdir -p RPMBUILD/{BUILD,RPMS,SOURCES,SRPMS,SPECS}
```

В каталоге **RPMS** также следует создать каталоги для каждой поддерживаемой архитектуры процессоров (x86_64, i386, noarch). Здесь каталоги создаются относительно корневого каталога **RPMBUILD** в целях удобства расположения.

Следующим этапом является создание файла спецификации (**.spec**), содержащего информацию о пакете, такую как имя, версия и описание, а также специальные инструкции для сборки. Создание файла спецификации является ключевым моментом процесса создания RPM пакета. Данный файл содержит информацию о конфигурации пакета, каким образом следует устанавливать файлы пакета, какие сценарии следует запускать до и после установки пакета. В следующем листинге приведен пример файла спецификации клиента **wget**:

```
# This is a sample spec file for wget
```

```

Summary:                GNU wget
License:              GPL
Name:                 wget
Version:             1.12
Release:            12
Group:               Development/Tools
Source:              %{name}-%{version}.tar.gz
BuildRoot:          %{_tmppath}/%{name}-%{version}
BuildArch:          i386
%description
The GNU wget program downloads files from the Internet using the command-line.
%prep
%setup -q
%build
./configure --prefix=$RPM_BUILD_ROOT/usr --sysconfdir=$RPM_BUILD_ROOT/etc
make
%install
make INSTROOT=$RPM_BUILD_ROOT install

%files
%defattr(-,root,root,-)
%config(noreplace) /etc/%{name}rc
%{_mandir}/man1/%{name}.1*
%{_bindir}/%{name}
%{_infodir}/*
%{_datadir}/locale/*/LC_MESSAGES/%{name}.mo
%doc AUTHORS COPYING ChangeLog MAILING-LIST NEWS README

```

По сути файл спецификации представляет собой набор директив и соответствующей им информации. Основные директивы приведенного примера (выделены жирным шрифтом) имеют следующее значение:

- **Summary** – Общее описание предназначения пакета.
- **Name** – Задаёт имя пакета (без пробелов).
- **Version** – Задаёт версию пакета, которая включает.
- **Release** – Задаёт номер релиза.
- **License** – Определяет тип лицензирования пакета, например GPL.
- **Group** – Определяет группу (см. приложение 2) программ к которой следует отнести данный пакет. Название группы должно присутствовать в файле `/usr/share/doc/rpm-<version>/GROUPS`.
- **Packager**, URL – Определяют субъекта (как правило, этим субъектом является человек, создавший пакет) и URL-адрес документации к пакету.
- **Source** – Задаёт имя архива (gzip или bzip2) с исходными кодами, находящегося в каталоге SOURCES. Если присутствует несколько архивов, то директивы нумеруются как Source0, Source1 и т.д. При задании имени файла рекомендуется использовать макросы `%{name}` и `%{version}` в целях универсальности..
- **BuildRoot** – Определяет расположение каталога, в котором будет выполняться сборка программ из исходного кода. При указании данной директи-

вы рекомендуется использовать макрос `%{tmppath}` для временного каталога, а также макросы, `%{name}` и `%{version}` для уникальности каталога, в котором будет выполняться сборка.

- **Requires** – Определяет пакеты, которые необходимо установить до установки данного пакета. Если необходимо выполнить дополнительные сборки программ из исходного кода, то указывается директива `BuildRequires`.
- **%description** – Задаёт описание пакета (`rpm -qi`).
- **%prep** – Содержит инструкции по распаковке архивных файлов с исходным кодом. Обычно выполняется макросом `%setup` в «скрытом» режиме.
- **%build** – Содержит описание набора команд необходимых для сборки программ из исходного кода (например, `make`).
- **%install** – Содержит описание набора команд необходимых для установки файлов программ (например, `make install`).
- **%clean** – Содержит описание набора команд необходимых для очистки каталога сборки от ненужных файлов.
- **%files** – Содержит список файлов, которые необходимо включить в пакет. Данная директива может содержать поддирективы, такие как `%doc`, `%defattr`, `%config`, `%attr` и прочие²¹.

После создания файла спецификации следует создать конфигурационный файл для сборщика – утилиты **make**. Данный файл содержит набор правил для сборки. Как правило, данный файл включен в архив с исходным кодом программы. Затем следует разместить файлы архивов с исходным кодом в каталоге **SOURCES**.

После размещения архива с исходным кодом выполняется сборка пакета при помощи утилиты **rpmbuild**:

```
rpmbuild -v -bb --clean RPMBUILD/SPECS/wget.spec
```

В данном примере ключ **-ba** указывает на то, что необходимо выполнить сборку бинарного RPM пакета и RPM пакета с исходным кодом; ключ **--clean** выполняет очистку сборочного каталога после создания пакета; ключ **-v** задаёт вывод на экран дополнительной информации в процессе создания пакета. Заключительным аргументом команды **rpmbuild** является файл спецификации, в данном случае – **RPMBUILD/SPECS/wget.spec**.

1.6. Организация хранения данных

Поскольку всё в ОС Linux представлено в виде файлов, от того, каким образом организовано их хранение файлов будет зависеть многое, в том числе производительность системы ввода-вывода, надёжность хранения и масштабируемость. Процесс организации хранения данных условно можно разделить на следующие этапы:

²¹ Описание дополнительных директив файла спецификаций доступно по адресу <http://www.rpm.org/max-rpm-snapshot/ch-rpm-inside.html>

- выбор необходимого устройства для хранения данных;
- создание разделов на данном устройстве и его форматирование (разметка);
- создание и настройка файловой системы на созданных разделах;
- пометка разделов и их монтирование в систему.

1.6.1. Создание и управление массивами RAID

Технология RAID позволяет администраторам создавать из нескольких жестких дисков единый логический диск, на котором данные распределяются или дублируются на несколько физических дисков. Существует два типа данной технологии: аппаратный и программный. В первом случае для организации RAID-массивов используются специальные устройства – RAID-контроллеры. Во втором случае все компоненты, необходимые для создания RAID-массивов, управляются операционной системой. В ОС Linux поддерживаются следующие уровни RAID-массивов.

- **Linear (конкатенация).** Массив, работающий в данном режиме, не обеспечивает избыточности данных или повышения производительности. Он просто объединяет два жестких диска, создавая при этом единый логический диск, равный сумме размеров дисков, из которых он был создан.
- **RAID 0.** Используется исключительно для повышения производительности. Записываемые данные параллельно распределяются между всеми дисками RAID массива.
- **RAID 1.** Данные дублируются одновременно на все диски RAID-массива. В этом режиме обеспечивается зеркальное дублирование данных, но производительность снижается, поскольку информация должна записываться более одного раза.
- **RAID 4.** Выполняется распределение данных между всеми дисками RAID-массива, но один из них выделяется для записи информации четности, что влечет за собой ожидание при записи на этот диск.
- **RAID 5.** Блоки данных и контрольные суммы циклически записываются на все диски RAID-массива, отсутствует выделенный диск для хранения информации о четности. При выходе из строя одного из дисков существенно снижается общая производительность.
- **RAID 6.** Хотя, как и в режиме RAID5, блоки данных и контрольные суммы циклически записываются на все диски RAID-массива, используется двойные контрольные суммы, поэтому производительность немного снижается, однако возможен отказ одного или двух дисков потери функционирования массива в целом.
- **RAID 10.** Данный режим сочетает распределение и дублирование данных, что оптимально с точки зрения производительности чтения и записи.

Для проверки того, какие уровни RAID можно использовать в установленной ОС Linux, необходимо просмотреть файл `/boot/config-<релиз_ядра>`, где вместо текстовой части `<релиз_ядра>` необходимо указать реальный релиз работающего ядра ОС Linux. В данном файле в директивах `CONFIG_MD_RAID*` должно быть указано значение «**m**».

По возможности лучше сконфигурировать RAID-массивы в процессе инсталляции ОС Linux. Если же существует необходимость в создании RAID-массивов после инсталляции, то необходимо установить пакет **mdadm**.

Для создания RAID-массива средствами ОС Linux необходимо последовательно выполнить следующие действия.

1. Добавить необходимое количество жестких дисков (зависит от выбранной конфигурации).
2. Создать разделы на данных дисках с типом **Linux raid auot**. В случае использования утилиты **fisk**, данный тип раздела имеет значение **fd**.
3. Создать необходимое количество RAID-массивов.
4. Настроить автоматическое монтирование созданных RAID-массивов в процессе загрузки ОС Linux.
5. Создать файловые системы на созданных RAID-массивах и добавить их описание в файл `/etc/fstab`.

Для создания разделов на дисках, которые будут использоваться под RAID-массив, используется команда **mdadm**, имеющая следующий синтаксис:

```
mdadm [режим] <устройство_RAID> [опции] <список_дисков> ,
```

- **[режим]** – параметр, в котором указывается, что необходимо сделать с RAID-массивом. Например, для того чтобы создать новый RAID-массив, необходимо указать ключ `-C` (`--create`). Полный список режимов работы с RAID массивами описан в руководстве `man` команды `mdadm`;
- **<устройство_RAID>** – полный путь к физическому устройству, которое будет являться RAID-массивом, например, `/dev/md0`;
- **[опции]** – различные опции, специфичные для каждого из доступных режимов работы. Основные опции, используемые в процессе создания RAID-массива, следующие:
 - **l** (`--level`) – используется для указания уровня RAID;
 - **n** (`--raid-devices=`) – количество активных дисков в RAID-массиве;
 - **x** (`--spare-devices=`) – количество дисков, которые будут использоваться для резервирования активных дисков;
 - **a** (`--auto`) – создавать автоматически устройство RAID при необходимости;
- **<список_дисков>** – общее количество дисков, включая активные и резервные диски.

Прогресс создания RAID-массива можно просмотреть, используя команду **tail -f /proc/mdstat**.

После того, как RAID-массив будет создан, необходимо внести соответствующие записи в файл `/etc/mdadm.conf`. Данный файл содержит следующие директивы, при помощи которых утилита **mdadm** управляет RAID-массивами.

- **DEVICE.** В данной директиве определяется, какие диски являются членами созданных RAID-массивов. Данная директива содержит список устройств, которые записываются через пробел, например: `DEVICE /dev/sda /dev/sdb`. В качестве устройств можно указывать как диски целиком, так и отдельные разделы. При указании устройств возможно использовать базовые регулярные выражения.
- **ARRAY.** В данной директиве выполняется идентификация массива с указанием входящих в его состав дисков. В данной директиве должен быть описан каждый RAID-массив; описание должно содержать такие параметры, как уникальный идентификатор (**UID**), список входящих в его состав устройств (**devices**), уровень RAID (**level**). Список параметров указывается через пробел.
- **MAILADDR.** Используется для отправки уведомлений в случае возникновения ошибок в работе RAID-массивов.
- **PROGRM.** Используется для запуска указанной команды для обработки событий мониторинга RAID-массивов.

Для формирования файла **mdadm.conf** можно воспользоваться командой **mdadm --detail --scan**, которая выполняет сканирование всех настроенных в системе RAID-массивов и выводит директиву **ARRAY**. После этого в данный файл необходимо добавить оставшиеся директивы и при необходимости добавить в директиву **ARRAY** дополнительные параметры.

После того, как был сконфигурирован файл `/etc/mdadm.conf`, необходимо создать на массиве файловую систему и настроить автоматическое монтирование данной файловой системы в файле `/etc/fstab`.

Часто необходимо осуществлять мониторинг состояния RAID-массивов. Для этого в ОС Linux существуют следующие возможности:

- просмотр файла `/proc/mdstat`, который содержит текущий статус RAID-массива;
- запуск команды **mdadm --query --detail /dev/mdN**, которая выводит более детальную информацию о массиве (**N** – номер устройства, например, `/dev/md0`);
- запуск команды **mdadm --examine <раздел>**, которая отображает детальную информацию о физическом устройстве, входящем в состав RAID-массива, (**<раздел>** обозначает физическое устройство, например, `/dev/sda1`).

1.6.2. Создание и управление логическими томами с помощью LVM

LVM (**Logical Volume Manager**) – это диспетчер логических томов, дополнительная подсистема, которая реализует особый вид разметки дисков на разделы. Диспетчер позволяет объединять отдельные диски в «**группы томов**». Суммарное пространство, занимаемое группой и рассматриваемое как единое целое, делится на логические тома, доступ к которым осуществляется как к обычным блочным устройствам (рис. 1.1.).

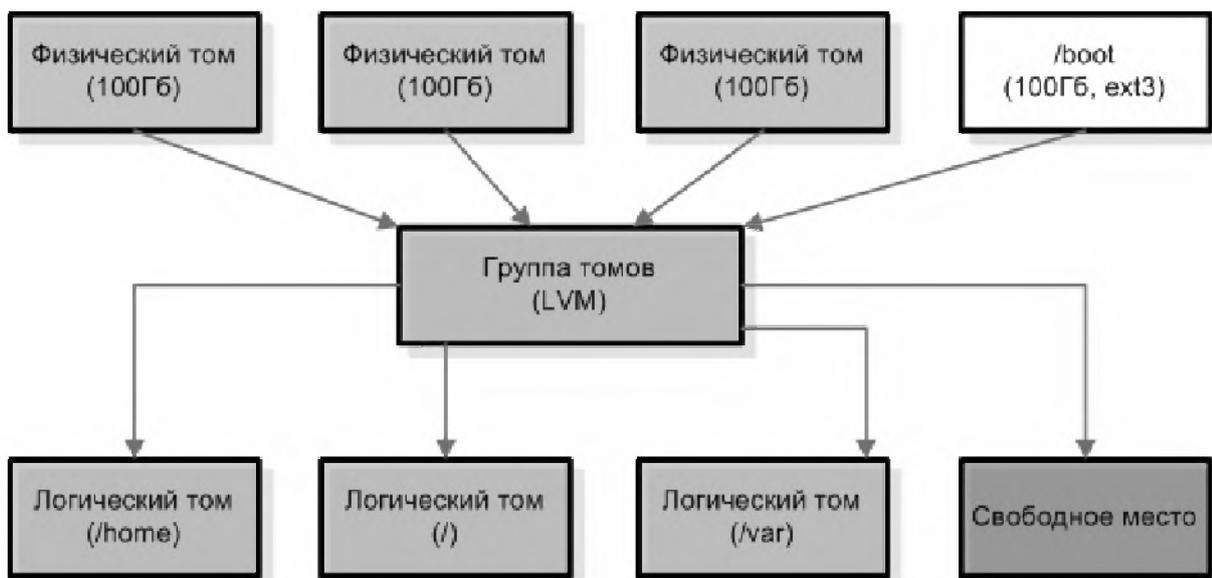


Рис. 1.1. Общая схема работы диспетчера логических дисков

Физические тома – это устройства, представляющиеся системе как один диск (жесткий диск или его раздел, RAID массив). **Группа томов** – это набор физических томов, объединенных в одно логическое устройство. **Логический том** – это логическое устройство, созданное на основе группы томов, которое содержит файловую систему.

Размер логических томов может впоследствии быть увеличен или уменьшен в зависимости от свободного места в группе томов. Раздел **/boot** не может быть создан на основе LVM, поскольку содержит ядро ОС Linux, загружающееся в память до того, как будет загружены соответствующие драйверы и определены группы LVM.

Благодаря технологии LVM возможно:

- более эффективно управлять дисковым пространством;
- перемещать логические тома между различными физическими устройствами;

- динамически изменять размеры логических томов;
- создавать снимки файловых систем;
- оперативно заменять диски без остановки системы.

Компоненты логического тома можно объединять различными способами. В **режиме конкатенации** физические блоки каждого диска остаются неразрывными, а диски как бы «перетекают» друг в друга. В **режиме чередования** соседние блоки виртуального диска могут на самом деле находиться на разных физических устройствах. Часто это позволяет повысить пропускную способность и уменьшить время, затрачиваемое на поиск данных.

Следует учитывать, что конфигурация LVM на этапе инсталляции ОС Linux возможна только в графическом режиме.

В ОС Linux имеются все средства для создания логических томов LVM после инсталляции. Данные средства входят в состав пакета **lvm**.

Создание логического тома LVM состоит из следующих этапов.

- 1) **Определение физических томов.** Для определения физического тома, который будет использоваться для хранения данных LVM, используется команда **pvcreate**. В качестве обязательного аргумента **pvcreate** используется название устройства, которое будет использоваться как физический том.
- 2) **Добавление физических томов в группу томов.** Для создания группы томов используется команда **pvcreate**, в качестве обязательных аргументов которой необходимо сперва указать желаемое название группы томов, а затем физический том, который будет использоваться для данной группы томов.
- 3) **Создание логических томов в группе томов.** Для создания логического тома используется команда **lvcreate**, имеющая следующий синтаксис:

```
lvcreate -n <имя_логического_тома> -l <количество_экстентов> <группа_томов>
```

- **<имя_логического_тома>** – желаемое название логического тома;
- **<группа_томов>** – название группы томов, в которой необходимо создать данный логический том;

Размер логического тома можно задавать как в стандартных единицах (Кб, Мб и т.д.), так и в **логических экстентах**, специальных блоках данных, которыми оперирует технология LVM. Для задания размера логического тома в стандартных единицах используется ключ **-L <размер>**.

- 4) **Создание файловой системы на логических томах и ее монтирование.**

Полный список команд управления устройствами LVM представлен в Приложении 3.

1.7. Резервное копирование и восстановление данных

Процедура резервного копирования сводится к резервированию файловой системы или ее отдельных файлов. Резервные копии позволяют администратору

восстанавливать файловую систему (или любую ее часть) в том состоянии, в каком она находилась на момент последнего копирования. Резервное копирование должно осуществляться тщательно и строго по расписанию. Кроме того, системы резервного копирования и сами носители резервных копий должны регулярно проверяться на предмет корректной работы.

В данном разделе мы рассмотрим стандартные утилиты резервного копирования, присутствующие в ОС Linux. К таким утилитам относятся **dump**, **restore**, **tar** и **rsync**. Помимо стандартных средств в ОС Linux также существует централизованная система резервного копирования **Amanda**²², позволяющая выполнять резервирование данных, находящихся на удаленных хостах с различными вариантами ОС Unix и ОС Windows.

Команды **dump** и **restore** являются штатным механизмом архивирования данных в ОС Linux. Для того чтобы воспользоваться данными командами, в системе должен быть установлен пакет **dump**.

1.7.1. Утилиты **dump** и **restore**

Команда **dump** позволяет делать резервные копии для файловых систем **ext2** и **ext3**. Команда формирует перечень файлов, которые изменялись с момента предыдущего архивирования, а затем упаковывает эти файлы в один большой архив, который записывается на внешнее устройство. Команда **dump** обладает рядом преимуществ по сравнению с другими утилитами, описанными в данном разделе:

- резервные копии могут быть записаны на несколько лент;
- можно выполнять резервное копирование и восстановление файлов любого типа (даже файлов устройств);
- права доступа, информация о владельцах и даты модификации файлов сохраняются;
- резервное копирование может выполняться в инкрементальном режиме (на ленту записываются только модифицированные версии файлов).

Команда **tar** также реализует все эти возможности, но средства инкрементального архивирования у команды **dump** немного лучше.

При создании архива ему присваивается номер уровня (целое число от 0 до 9). В архив уровня **N** копируются все файлы, которые изменились с момента создания последнего архива уровня ниже **N**. В архив нулевого уровня включается вся файловая система. В режиме инкрементального архивирования файловую систему можно вернуть в то состояние, в котором она находилась в момент последнего резервного копирования, хотя для этого может потребоваться восстановление файлов с нескольких носителей.

Первым аргументом команды **dump** должен быть уровень архива. Для того чтобы определить, когда создавался последний архив, команда **dump** проверяет файл `/etc/dumpdates`. Ключ **-u** заставляет команду по завершении копирования

²² Информации о системе резервного копирования **Amanda** доступна по адресу <http://www.amanda.org/>.

автоматически обновить файл `/etc/dumpdates`. При этом регистрируются дата, уровень архива и имя файловой системы. Если флаг `-u` ни разу не использовался, всем архивам будет присваиваться уровень 0, поскольку факт предыдущего резервного копирования файловой системы нигде не был зафиксирован.

Свою выходную информацию команда **dump** посылает на устройство, заданное по умолчанию. Для указания другого устройства необходимо использовать ключ `-f`. Большинство современных ленточных накопителей автоматически сообщают команде **dump** о длине ленты. В противном случае длину ленты необходимо указать с помощью опции `-B`. Последним обязательным аргументом команды **dump** является точка монтирования или список файлов, например, `/home`.

Пример использования команды **dump** для резервирования каталога `/home` в файл `/opt/backup`:

```
dump -0u -f /opt/backup/users_homes.dump /home
```

Для восстановления данных, зарезервированных командой **dump**, используется команда **restore**. Наиболее важными опциями команды **restore** являются: `-i`, позволяющая восстанавливать отдельные файлы и каталоги в интерактивном режиме, и `-r`, задающая полное восстановление всей файловой системы. Опция `-x` запрашивает автоматическое восстановление указанных файлов.

Команда **restore -i** считывает из резервной копии таблицу содержимого, а затем позволяет перемещаться по архиву, как по обычному дереву каталогов, с помощью команд **ls**, **cd** и **pwd**. Найдя файл, который нужно восстановить, необходимо выполнить команду **add <имя_файла>**. Когда все необходимые файлы выбраны, необходимо извлечь их при помощи команды **extract**. Восстановление файлов данным методом лучше производить, в отдельном разделе, и лишь после восстановления перенести восстановленный файл в нужное место.

Если на одной ленте находится несколько архивов, то перед выполнением команды **restore** необходимо перемотать ленту на соответствующий архив с помощью команды **mt**.

1.7.2. Утилита tar

Утилита **tar** входит в состав ОС Linux по умолчанию и имеет достаточно гибкие настройки для резервирования данных. Данная команда объединяет несколько файлов или каталогов в один файл, который затем можно сжать и записать на резервный носитель. Это удобный инструмент создания резервных копий файлов, которые предполагается восстанавливать в ближайшем будущем.

Для того чтобы создать сжатый архив с компрессией **bzip2** используется следующая команда:

```
tar cjvf <имя_архива>.tar.bz <файл1> <файл2> ...
```

В данном примере для создания архива используется аргумент **c**. Аргумент **j** используется для указания типа компрессии. Аргумент **v** используется для более детального вывода работы команды, а аргумент **f** – для указания создаваемого файла архива. После указания имени архива необходимо перечислить все файлы,

которые будут в него помещены.

Для того чтобы извлечь данный файл в текущий рабочий каталог, необходимо запустить команду **tar** с теми же аргументами, но вместо аргумента создания архива (**c**) необходимо указать аргумент извлечения данных из архива (**x**):

```
tar xjvf <имя_архива>.tar.bz
```

При извлечении файлов из архива сохраняется исходная структура каталогов. Для просмотра содержимого архива без распаковки необходимо выполнить команду **tar** следующим образом:

```
tar tjvf <имя_архива>.tar.bz
```

При помощи команды **tar** можно перемещать деревья каталогов, особенно если файлы копируются от имени пользователя **root**. Она сохраняет информацию о принадлежности объектов и времени их создания, но только если это указано в ее параметрах.

Для создания копии дерева исходного каталога в целевом каталоге можно выполнить команду:

```
tar -cf - <исходный_каталог> | (cd <целевой_каталог>; tar --atime-preserve -xpf -)
```

По умолчанию команда **tar** не выполняет разрешение символических ссылок, но может делать это при указании опции. Можно также включать в архив только те файлы, которые были модифицированы после заданной даты, что очень удобно для организации инкрементального резервного копирования.

1.7.3. Утилита rsync

Утилита **rsync** широко используется в ОС Linux для копирования файлов на удаленные хосты, а также для синхронизации между локальными каталогами. Кроме того, большинство файлообменных сервисов в сети Интернет поддерживает протокол **rsync** для загрузки или скачивания файлов.

Основным преимуществом команды **rsync** при копировании файлов является то, что копируются только изменения, сделанные в файле, тем самым существенно уменьшается общее время копирования. Кроме того, утилита **rsync** позволяет сжимать данные в процессе передачи и осуществлять надежный контроль целостности данных.

Например, для копирования домашнего каталога пользователей **/home** в каталог **/backups**, находящийся на удаленном сервере **backup.linux.lab** используется следующая команда:

```
rsync -avz /home backup.linux.lab:backups/
```

Здесь аргумент **-a** используется для архивирования данных, т.е. выполняется рекурсивное копирование всех подкаталогов вместе с их файлами с сохранением прав доступа, символьных ссылок, временных отпечатков файлов, владельца и группы файлов. Аргумент **-z** используется для сжатия данных перед копированием.

При копировании файлов на удаленный хост важно наличие или отсутствия символа **</>** в конце названия копируемого каталога. Если символ **</>** в конце

названия копируемого каталога отсутствует, на удаленный хост будет создан аналогичный каталог со всем его содержимым. Если данный символ указан, то на удаленном хосте будут скопированы только файлы, содержащиеся в копируемом каталоге.

Утилита **rsync** может работать в режиме «клиент-сервер». В данном случае на сервере функционирует демон **rsyncd**, который ссылается на конфигурационный файл **/etc/rsyncd.conf**. Файл **/etc/rsyncd.conf** содержит глобальные конфигурационные параметры и секции с описанием модулей, каждая из которых представляет собой дерево каталогов для экспорта или импорта:

```
#Название модуля (используется при подключении в командной строке)
[upload]
# Каталог, в который будет производиться загрузка файлов
path = /var/rsync/pub
# Режим доступа к модулю (чтение или чтение+запись)
read only = false
# Идентификаторы пользователя и группы, которые будут использоваться при загрузке
файлов
uid = root
gid = root
# Список хостов, которым разрешено загружать файлы в данный модуль
hosts allow = main.linux.lab
```

В данной конфигурации все операции выполняются в каталоге **/var/rsync/pub**, а доступ разрешен только хосту **main.linux.lab**. С точки зрения пользователя или клиента можно осуществлять загрузку файлов на сервер, указывая в качестве пункта назначения выражение вида **<хост>::<модуль>**, которое соответствует описанному выше модулю.

1.8. Автоматизация выполнения задач

Очень часто администратору приходится выполнять повторяющиеся задачи: ротацию системных журнальных файлов, проверку ресурсов оборудования, резервное копирование и др. Повторяющиеся задачи можно выполнять при помощи имеющегося в ОС Linux планировщика задач **cron**. Для использования планировщика задач **cron** в ОС Linux должны быть установлены пакеты **vixie-cron** и **crontabs**.

Планировщик задач **cron** присутствует в системе в виде фонового процесса **crond**, который каждую минуту считывает несколько конфигурационных файлов, содержащих списки командных строк и расписание их вызова. Командные строки выполняются в командном интерпретаторе **bash**.

Для управления конфигурационными файлами планировщика **cron** используется команда **crontab**. Основной конфигурационный файл, используемый для задания системных административных заданий, **/etc/crontab**, содержит следующие записи:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

```
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

В первых четырех строках данного листинга определяются переменные окружения, которые будут доступны для заданий планировщика cron. Каждая строка в файле **/etc/crontab** состоит из шести полей: минута, час, число (день месяца), день недели, команда.

Каждое из полей времени может содержать:

- символ «*», обозначающий любую цифру;
- целое число;
- два целых числа, разделенные символом «-», обозначающих диапазон;
- целые числа или диапазоны, разделенные запятыми.

Диапазоны значений могут содержать значение шага поиска. Например, последовательность 0,3,6,9,12,15,18 можно записать короче: 0-18/3. Можно также использовать сокращенные названия месяцев и дней недели, но только не вместе с диапазонами значений.

Все остальные символы в строке файла **crontab** интерпретируются как выполняемая команда с ее параметрами. Если команда отправляет какой-нибудь текст в стандартный канал вывода, этот текст отправляется по почте пользователю, указанному в переменной **MAILTO**.

В файле **etc/crontab** при помощи bash-скрипта **run-parts** запускаются команды, содержащиеся в каталогах **/etc/cron.hourly**, **/etc/cron.daily**, **/etc/cron.weekly** и **/etc/cron.monthly**. Из названий каталогов понятно, что запуск выполняется каждый час, каждый день, каждую неделю и каждый месяц. Как правило, в каталогах **/etc/cron.*** содержатся системные команды, выполняющие периодическую работу. Например, в каталоге **/etc/cron.daily** содержатся скрипты утилиты **logrotate**, выполняющей ротацию журнальных файлов, скрипт утилиты **updatedb**, которая обновляет базу данных языковых параметров (locale), скрипт **tmpwatch**, который периодически удаляет файлы из каталогов **/tmp** и **/var/tmp**.

Помимо выполнения административных задач, у обычных пользователей существует также возможность задавать пользовательские задания. Для этого пользователю необходимо выполнить команду **crontab -e**, в результате которой откроется его персональный файл расписания, **/var/spool/cron/<регистрационное_имя_пользователя>**, в котором пользователь задает расписание выполнения команд. Для каждого пользователя предусмотрен только один такой файл. В качестве имени файла выбирается регистрационное имя пользователя, которому он принадлежит, и на основании этого имени демон **crond** выясняет, какое значение UID нужно использовать при выполнении команд, содержащихся в данном файле.

В ОС Linux существует возможность предоставлять доступ к демону **crond** только определенным пользователям. Два файла конфигурации: **/etc/cron.deny**

и `/etc/cron.allow` содержат информацию о том, кому из пользователей разрешено использовать команду `crontab`. Если файл `cron.allow` существует, то он содержит список пользователей, имеющих доступ к демону `crond` (по одному имени в строке). Пользователи, которых нет в списке, не имеют права выполнять команду `crontab`. Если файла `cron.allow` нет, проверяется файл `cron.deny`. Он также содержит список пользователей: доступ разрешен всем, кроме указанных в данном файле. Если нет ни одного из этих файлов, использовать команду `crontab` имеет право только пользователь `root`.

При запуске демон `crond` считывает все свои конфигурационные файлы, помещает их в память и переходит в режим ожидания. Раз в минуту демон активизируется, проверяет даты модификации пользовательских и системных расписаний и загружает те файлы, которые изменились, а затем выполняет команды, запланированные на данную минуту, после чего снова переходит в режим ожидания.

Помимо планировщика задач `cron` в ОС Linux существует планировщик `at`, также предназначенный для выполнения команд по расписанию, однако этот планировщик может быть настроен только на единовременное выполнение команды в заданное время. Планировщик `at` работает по схожей схеме, что и `cron`: запланированные задания помещаются в очередь в каталоге `/var/spool/at` и выполняются в указанное время. Для запланированного запуска заданий используется команда `at`, имеющая следующий синтаксис:

```
at [-f <файл>] <время>
```

При указании аргумента `-f` с параметром `<файл>` команда `at` будет брать задания из указанного файла. В аргументе `<время>` указывается время, когда должно быть запущено задание. Если аргумент `-f` опущен, то задания считываются из стандартного канала ввода.

Таблица 1.11. Примеры запуска команды `at`

Временной период	Пример	Описание
Минуты	<code>at now + 10 minutes</code>	Задания запускаются через десять минут
Часы	<code>at now + 2 hour</code>	Задания запускаются через час
Дни	<code>at now + 1 day</code>	Задания запускаются через 24 часа
Недели	<code>at now + 1 week</code>	Задания запускаются через 7 дней
–	<code>at 20:00 31/12/08</code>	Задания запускаются в 20:00 31 декабря 2008 года

Для просмотра очереди заданий демона `atd` используется команда `atq`.

1.9. Журналирование системных событий

Неотъемлемой частью контроля и мониторинга работы ОС Linux и всех ее сервисов является журналирование событий. Все утилиты создают журнальные

Уровень	Значение
err	Сообщения об ошибках
warning	Предупреждающие сообщения
notice	Сообщения, имеющие интерес
info	Информационные сообщения
debug	Отладочные сообщения

Например, предположим, что в файле **syslog.conf** задана следующая строка:

```
*.info;mail.none;news.none;authpriv.none;cron.none /var/log/messages
```

В данной строке осуществляется посылка сообщений, удовлетворяющих указанному фильтру, в файл **/var/log/messages**, а именно:

- посылка всех сообщений, имеющих уровень важности **info** и выше;
- отмена посылки всех событий для средств **mail**, **news**, **authpriv** и **con**.

В файле **syslog.conf** уровни обозначают *минимальную* важность, которую сообщение должно иметь для того, чтобы быть зарегистрированным. Например, сообщение почтовой системы, имеющее уровень важности **warning**, будет соответствовать фильтру **mail.warning**, а также фильтрам **mail.info**, **mail.notice**, **mail.debug**, ***.warning**, ***.notice**, ***.info** и ***.debug**. В ОС Linux возможно ограничивать посылку сообщений только определенных уровней. Для этого используются специальные символы «=» и «!», обозначающие «только данный уровень» и «кроме данного и более высоких уровней».

Таблица 1.13. Примеры использования ограничений посылки событий

Обозначение	Значение
mail.info	Посылка сообщений почтовой системы с уровнем info и выше
mail.=info	Посылка сообщений почтовой системы только с уровнем info
mail.info;mail.!err	Посылка сообщений почтовой системы с уровнями info, notice и warning
mail.debug;mail.!=warning	Посылка сообщений почтовой системы с любым уровнем, кроме warning

Таблица 1.14. Действия syslog

Действие	Значение
имя_файла	Записать сообщение в указанный файл на локальном хосте
@имя_хоста	Переслать сообщение демону syslogd, функционирующему на удаленном хосте с указанным именем

Действие	Значение
@IP_адрес	Переслать сообщение демону syslogd, функционирующему на удаленном хосте с указанным IP-адресом
файл_FIFO	Переслать сообщение в указанный именованный канал
пользователь1, пользователь2,...	Переслать сообщение на экран терминала указанных пользователей, если они зарегистрированы в системе
*	Переслать сообщения на экраны всех зарегистрированных в системе пользователей

Журналирование событий осуществляется в регулярные файлы. При необходимости перед файлом можно указать знак «-», чтобы не использовать синхронизацию данных в файле после каждой посылки событий. Это может немного повысить производительность в случае, если приложение использует очень детализированный уровень важности.

1.10. Мониторинг и оптимизация производительности

Производительность ОС Linux во многом определяется эффективностью распределения и совместного использования ее основных ресурсов. В первом приближении основное влияние на производительность оказывают следующие факторы:

- время использования центрального процессора;
- использование виртуальной памяти;
- пропускная способность дисковой подсистемы ввода-вывода;
- пропускная способность сетевой подсистемы ввода-вывода.

1.10.1. Процессор

Самый простой с точки зрения учета ресурс – время использования центрального процессора (CPU). В распоряжении системы всегда находится практически 100% его мощности. Для процесса, занимающего более 90% времени ЦП, ограничивающим фактором является быстродействие центрального процессора. Такой процесс потребляет большую часть вычислительных мощностей системы. Основными параметрами, влияющими на производительность центрального процессора являются:

- количество задач, ожидающих выполнения;
- утилизация процессора;
- приоритет выполнения процессов.

Для того, чтобы просмотреть количество задач, решаемых процессором, используется команда **uptime**. Данная команда отображает общее время работы системы, количество работающих пользователей и нагрузку на процессор (**load average**). Вывод данной команды имеет следующий вид:

```
05:08:07 up 6:10, 2 users, load average: 0.00, 0.00, 0.00
```

В идеале количество задач на процессоре должно быть равным 1, что говорит о том, что каждый процесс имеет постоянный непрерывный доступ к процессору. В действительности из-за большого количества задач данный параметр может находиться в пределах от 8 до 10 и даже выше. Большая утилизация процессора (время использования его ресурсов) также не всегда является показателем падения производительности, поскольку процессор может ожидать получения необходимых данных из других подсистем, например, получения данных из виртуальной памяти.

Пользователь не имеет возможности изменять приоритет процессов, он может изменять только приоритет предпочтения (**nice**) процессов. Если какие-то процессы работают медленно, можно предоставить им больше процессорного времени, снижая их приоритет предпочтения (**nice priority**). Например, для увеличения приоритета предпочтения для работающего процесса на -5 необходимо использовать следующую команду:

```
renice -5 <PID>
```

По умолчанию пользовательские процессы имеют приоритет предпочтения, равный 0. Установка отрицательного значения для данного параметра позволяет процессу использовать большую долю времени работы процессора.

Для того чтобы определить утилизацию процессора, приоритет предпочтения и реальный приоритет выполнения процессов, используются команды **top** или **ps**.

Если в системе используется несколько процессоров, то имеет смысл сгруппировать процессы, которые вызывают большое количество прерываний процессоров, на одном или нескольких процессорах – это позволит избежать перемещения процессов между процессорами и достигнуть более оптимальной работы процессорного кэша. Данный механизм оптимизации заключается в организации «родственности процессов» (**CPU affinity**) по отношению к процессорам. Например, для того, чтобы какое-либо прерывание выполнялось на 4-ом процессоре, необходимо задать значение маски процессора в файле **smp_affinity**:

```
echo 03 > /proc/irq/19/smp_affinity
```

Общую информацию об использовании процессора можно узнать с помощью команды **vmstat**. Она принимает два аргумента: время (в секундах), в течение которого необходимо наблюдать за системой для получения каждой строки выходной информации, и необходимое количество замеров. Если не указать число замеров, команда будет выполняться до тех пор, пока пользователь не нажмет клавиши **<Ctrl+C>**.

В первой строке отображаемых данных сообщаются средние значения, измеренные с момента запуска системы. В следующем листинге приведены результаты

каждого очередного замера, стандартная продолжительность которого составляет 5 секунд.

```
# vmstat 5 5
procs -----memory----- ---swap-- -----io----- --system-- ---cpu--
r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0 0    0 73464 69748 163304  0  0   7  7 1005 50  0  0 99  0  0
0 1    0 73464 69752 163300  0  0   0  5 1003 44  0  0 100  0  0
0 0    0 73464 69756 163304  0  0   0  1 1004 44  0  0 100  0  0
0 0    0 73464 69756 163304  0  0   0  0 1003 43  0  0 100  0  0
0 0    0 73464 69756 163304  0  0   0  0 1004 44  0  0 100  0  0
```

В результате отображается ряд параметров: пользовательское время (**us**), системное время (время ядра) (**sy**), время простоя (**id**) и время ожидания для каналов ввода-вывода (I/O) (**wa**). Большие числа в колонке **us** обычно означают вычисления, а в колонке **sy** они свидетельствуют о том, что процессы осуществляют очень много системных вызовов или выполняют операции ввода-вывода. Общее правило распределения времени процессора заключается в следующем: система должна тратить примерно 50% рабочего времени на обслуживание пользовательских запросов и столько же – на системные запросы. Если сервер выделяется под одно-единственное, но интенсивно использующее процессор приложение, большая часть времени должна тратиться на обработку пользовательских запросов.

В колонке **cs** показано число переключений контекста (**context switches**) за данный период, т.е. сколько раз планировщик процессов (**CPU scheduler**) переключал процессы. В колонке **in** отображается число прерываний, генерируемых аппаратными устройствами или компонентами ядра. Слишком большие значения в этих колонках свидетельствуют о неправильно работающем аппаратном устройстве. Остальные колонки важны для анализа использования памяти и жесткого диска.

Как видно из показанного примера, центральный процессор практически простаивает, о чем свидетельствует среднее время простоя (**id**), равное 100%.

В многопроцессорных системах команда **vmstat** выводит среднее значение по всем процессорам. Для вывода информации по конкретному процессору необходимо использовать команду **mpstat**. Данную команду удобно использовать при отладке программ, поддерживающих многопроцессорную²⁴ обработку.

Общие рекомендации по оптимизации производительности использования процессора заключается в следующем.

- Используя команду **renice**, определите не критичные для вас процессы, работающие в системе, и снизьте их приоритет выполнения.
- При помощи команды **ps** определите запущенные в системе процессы, которые реально не используются. Если такие процессы существуют, определите какие сервисы их запускают и отключите их запуск.
- В многопроцессорной системе следует разделить все процессы между раз-

²⁴ Обычно положительный эффект, с точки зрения производительности, от использования многопроцессорных систем могут получить приложения, использующие технологию **multi threading**.

ными процессорами, используя команду **taskset**. Это позволит не допустить частого очищения процессорных кешей, которое происходит при переходе процесса от одного процессора к другому.

- Используя команду **top**, определите динамику изменения утилизации процессора различными процессами. Это поможет понять, какие процессы максимально используют вычислительные ресурсы процессора.
- Используя команду **vmstat**, определите общие показатели работы процессора. Если время простоя процессора практически равно нулю, необходимо разобраться, какие процессы используют процессор и задуматься о его модернизации, так как для стабильной работы системы должен существовать хотя бы небольшой запас процессорного времени, хотя бы 10-15%.
- Назначьте приоритет для прерываний, которые чаще других прерывают процессор, используя утилиту **irqtune** и файл **/proc/interrupts**.

1.10.2. Память

В ОС Linux организация доступа к памяти реализована с использованием механизмов *виртуальной памяти*. Каждому процессу в ОС Linux выделяется некоторое виртуальное адресное пространство. При использовании 32-битной архитектуры процессора каждому процессу доступно до 4 Гб виртуального адресного пространства. При этом по умолчанию 1 Гб этого всего виртуального адресного пространства отдается ядру. Таким образом, пользователю становятся доступны только 3 Гб памяти. При использовании 64-битной архитектуры процессора данное ограничение отсутствует, и пользовательским процессам доступно практически неограниченное количество виртуального адресного пространства.

Приложения, работающие в ОС Linux, ничего не знают об архитектуре физической памяти. Вся физическая память транслируется в виртуальную (**memory mapping**). Виртуальная память может быть транслирована не только в физическую память – если приложению требуется большее количество памяти (больше, чем общее количество физической памяти), то некоторая часть виртуальной памяти транслируется в пространство подкачки (**swap space**) дисковой подсистемы.

Виртуальная и физическая память в ОС Linux организована в виде модулей, называемых *страницами*. Размер одной страницы, как правило, составляет 4 Кбайт. Каждая такая страница транслируется на физическое хранилище – ОЗУ или жесткий диск. Информация о связях между виртуальными и реальными страницами хранится в *таблице страниц*.

Ядро ОС Linux способно эффективно обрабатывать запросы процессов, дополняя физическую память областью подкачки. Системе постоянно приходится перемещать страницы между ОЗУ и областью подкачки. Такие операции называются *страничным обменом* (**paging**).

Управление страницами осуществляется так, чтобы те из них, к которым обращение было сделано недавно, хранились в физической памяти, а менее активные выгружались на диск. ОС Linux отслеживает частоту обращения к каждой

странице виртуальной памяти. Ядро ведет несколько списков страниц. Страницы, частота обращения которых больше нуля, считаются активными и помещаются в список активных страниц (**active pages**). Всякий раз, когда происходит обращение к странице, ее частота обращения увеличивается. Параллельно с этим выполняется демон **kswapd**, который регулярно уменьшает частоту обращения неактивных страниц. Как только частота обращения снижается до нуля, демон **kswapd** перемещает страницу в список неактивных страниц (**inactive pages**) и удаляет ссылку на нее из таблицы страниц. Такие страницы считаются готовыми к выгрузке на диск, однако при поступлении запроса к ним ядро сможет восстановить их из памяти или с диска и снова поместить в таблицу страниц. Страницы, выгруженные на диск, должны быть прочитаны с диска, прежде чем их можно будет использовать. Количество активных и не активных страниц можно просмотреть, используя команду **vmstat -a**.

Когда ядру не хватает как физической памяти, так и области подкачки, это означает, что виртуальная память исчерпана. В данной ситуации включается режим «принудительного уничтожения» процессов. Чтобы освободить память, системе приходится уничтожать целые процессы.

Интенсивность операций с памятью количественно представляется общим объемом задействованной виртуальной памяти и скоростью подкачки и выгрузки страниц. Определить объем доступной памяти и размер области подкачки можно при помощи команды **free**. При введении ключа **-t** будет выведен суммарный объем виртуальной памяти. В поле **free** вывода команды **free** указывается объем (в килобайтах) списка неиспользуемых страниц. Если приведенные в нем показатели менее 3% от общего объема системной памяти, то это свидетельствует о наличии проблем. В поле **total** указано общее количество доступных страниц (в килобайтах). В полях **buffers** и **cached** указано количество страниц (в килобайтах), занятых под буферы ядра и под дисковый кеш. В поле **used** указано количество страниц (в килобайтах), используемых приложениями. Вывод, отображаемый командой **free**, является статическим снимком с текущего использования памяти. Для отображения нескольких снимков используется команда **free -s <интервал>**, где указывается интервал вычислений в секундах.

Для более детальной статистики использования памяти можно воспользоваться командой **vmstat**, которая выдает информацию о страничном обмене и областях подкачки. Существует еще один источник информации об использовании памяти – команда **procinfo**, входящая в состав пакета **procinfo**. Данная команда самостоятельно не ведет никаких замеров и вычислений, а просто форматирует данные, извлекаемые из элементов файловой системы **/proc**. Если нужно получать отчет, обновляющийся, например, каждые 5 секунд, необходимо использовать команду **procinfo -n5**.

Основные рекомендации по оптимизации производительности использования ресурсов виртуальной памяти, заключается в следующем:

- Увеличивать или уменьшать размер страниц виртуальной памяти в зависимости от приложения, которое использует ресурсы памяти.

- Оптимизировать процесс обработки активных и неактивных страниц виртуальной памяти.
- Снизить частоту выгрузки страниц с жесткого диска в память.
- Ограничить ресурсы виртуальной памяти для каждого пользователя в системе, используя команду **ulimit**.
- Остановить процессы, которые реально не используются в системе.

1.10.3. Подсистема дискового ввода-вывода

Производительность жесткого диска можно оценить при помощи команды **iostat**. Данная команда поддерживает дополнительные аргументы, задающие интервал времени в секундах между моментами выдачи результатов за истекший период и число повторений замеров. Команда выдает также сведения об использовании процессора. В следующем листинге приведен пример использования команды **iostat** для определения производительности дискового ввода-вывода.

```
# iostat
```

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	0.93	8.91	10.77	368451	445574
sdb	0.00	0.02	0.00	864	80
sdf	0.03	0.02	1.92	1016	79318
sdg	0.03	0.03	1.92	1364	79318
hdc	0.00	0.00	0.00	144	0
md0	0.00	0.03	0.00	1076	10
md1	0.96	0.03	1.91	1092	79126
dm-0	0.00	0.01	0.00	608	128
dm-1	0.96	0.01	1.91	548	79126
dm-2	0.00	0.01	0.00	532	10

Команда **iostat** собирает данные из файловой системы **/proc**, выдавая по одной строке для каждого физического устройства. Для каждого жесткого диска сообщается число операций ввода-вывода в секунду (**tps**), количество операций блочного чтения и блочной записи в секунду (**Blk_read/s** и **Blk_write/s**), общий объем прочитанных (**Blk_read**) и записанных (**Blk_wrtn**) блоков.

Для получения более детальной информации о конкретном устройстве необходимо запустить команду **iostat** следующим образом:

```
# iostat -dx /dev/sda
```

```
rrqm/s wrqm/s r/s w/s rsec/s wsec/s avgrq-sz avgqu-sz await svctm %util
0.11 0.73 0.32 0.58 8.58 10.48 21.06 0.01 9.40 3.57 0.32
```

В данном примере к устройству **/dev/sda** было сделано 0,58 обращений на запись данных в секунду (**w/s**). Среднее значение запроса к устройству составило 21,06 блоков (**avgrq-sz**), при этом утилизация процессора при выполнении данной операции (**await**) составила 9,4%. Среднее время обслуживания (**svctime**) обращений к устройству **/dev/sda** составило 3,57 миллисекунды. Основное внимание здесь следует уделить параметру времени обслуживания обращений (**svctime**),

которое существенно зависит от времени поиска нужных блоков.

Разбивая новый жесткий диск на разделы, необходимо принять во внимание факторы, влияющие на производительность, и постараться поместить файлы, обращение к которым осуществляется одновременно, в одну файловую систему. Для достижения максимальной производительности нужно помещать файловые системы, которые используются одновременно, на физически разные диски. Особенно важно распределить область подкачки между несколькими дисками, если это возможно, поскольку страничный обмен обычно замедляет работу системы в целом. Такую конфигурацию можно реализовать с помощью команды **swapon**.

Основные рекомендации по оптимизации производительности использования дисковой подсистемы сводятся к следующему.

- Если основная нагрузка на диск носит последовательный характер, рекомендуется увеличить кеш дискового контроллера или размещать данные, к которым осуществляется последовательный доступ, на одной файловой системе, а лучше – на смежных блоках. Если основная нагрузка носит случайный характер, то увеличение количества дисков может повысить общую производительность.
- Для повышения операций чтения и записи следует по возможности использовать аппаратные массивы RAID.
- При необходимости создания разделов большого размера рекомендуется соединять отдельные диски в единый логический том LVM, вместо того чтобы использовать один диск большого размера.
- Добавлять большее количество физической памяти для увеличения дискового кэша.

1.10.4. Комплексные средства мониторинга системных ресурсов

В ОС Linux помимо отдельных утилит мониторинга системных ресурсов существуют средства, позволяющие накапливать статистическую информацию и строить соответствующие отчеты. Одним из таких средств является утилита **sar**. На первый взгляд может показаться, что она отображает ту же информацию, что и утилиты **vmstat** и **iostat**. Однако имеется одно очень важное отличие: утилита **sar** имеет возможность представлять отчеты как по накопленным, так и по текущим данным.

Для использования данной утилиты необходимо установить пакет **sysstat**. Данные, которые накапливает утилита **sar**, могут собираться через определенный интервал времени, предоставляя тем самым администратору полную картину производительности в течение интересующего его периода.

Прежде чем использовать утилиту **sar**, необходимо инициализировать дополнительные утилиты **sa1** и **sa2**, которые используются для сбора и записи статистических данных. Данные утилиты автоматически запускаются при помощи планировщика задач **cron** согласно расписанию, заданному в файле `/etc/cron.d/sysstat`.

По умолчанию утилита **sar** формирует отчеты каждый день в 23:53 и сохраняет их в каталоге `/var/log/sa` в бинарном виде, используя имена файлов в формате **sa<дата>**, где вместо параметра **<дата>** указывается двухзначное число месяца. В формате отчетов не используется отметка года, поскольку в каталоге **sa** сохраняются только последние 9 отчетов. Для просмотра отчетов используется команда **sar**, которая может предоставлять информацию как об использовании процессорного времени, так и об активности дисковой подсистемы, а также статистические данные о сетевых интерфейсах.

Вывод команды **sar** без параметров отображает статистику использования процессорного времени и имеет следующий вид. Для просмотра дополнительных аргументов команды **sar** можно воспользоваться страницами руководства **man** по данной команде.

1.11. Установка программ из исходного кода

В установленном дистрибутиве ОС Linux не всегда может присутствовать нужное программное обеспечение. В данном случае может понадобиться установить новый пакет, однако пакеты, доступные в сети RHN или любом другом репозитории, могут более ранних версий, чем на сайте разработчиков, и не содержать требуемый функционал. Тогда возникает необходимость установки ПО из исходного кода, доступного на сайте разработчиков.

Установка ПО из исходного кода может быть полезна и в случае необходимости оптимизации программного кода под имеющееся оборудование. Для того чтобы установить ПО из исходного кода прежде всего необходимо загрузить нужную версию с ресурса разработчиков данного ПО. Как правило, исходный код распространяется в архивных файлах в форматах **tar.gz** или **tar.bz2**. В данных архивах распространяются в основном стабильные версии программ. Если необходимо получить самые последние изменения кода, то стоит воспользоваться хранилищами версий²⁵, которые представляют собой репозитории, используемые разработчиками как средство совместной разработки и обмена кодом. Доступ к таким хранилищам возможен при помощи специальных утилит (в зависимости от типа хранилища) или через веб-интерфейс.

После того, как исходный код будет получен, его необходимо скомпилировать при помощи компилятора и установить в систему. Для удобства управления программами, созданными из исходного кода, рекомендуется создавать на их основе **rpm** пакеты.

В среде ОС Linux основным инструментом сборки программ из исходного кода является набор компиляторов GCC (GNU Compiler Collection). Данные компиляторы являются кросс-платформенными и позволяют компилировать исходный код, написанный на таких языках программирования как C/C++, Objective-

²⁵ Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

C, Java, Fortran, и Ada. Помимо компилятора, для сборки программ из исходного кода необходимы дополнительные утилиты, входящие в так называемый набор **GNU toolchain** – коллекция утилит GNU для сборки программ из исходных кодов. в состав GNU toolchain входят следующие инструменты:

- **GCC** – Набор компиляторов для различных языков программирования;
- **GNU make** – Утилита автоматизации процесса компиляции и сборки программ;
- **GNU Binutils** – Набор утилит, включая компоновщик (ld), ассемблер (as) и другие утилиты;
- **GNU Bison** – Утилита автоматизации создания синтаксических анализаторов;
- **GNU m4** – Макропроцессор²⁶;
- **GNU Debugger (GDB)** – Универсальный отладчик программ;
- **GNU autotools** – Утилиты (autoconf, automake, autoheader, libtool)) автоматизации конфигурации пакетов с исходным кодом;
- **GNU libc** – Стандартная библиотека языка C. Все программы, написанные на языке C, опираются на функции стандартной библиотеки.

Для установки набора GNU toolchain необходимо установить соответствующие пакеты RPM или собрать его из исходного кода. В общем случае последовательность установки программы из исходного кода следующий:

- 1) Конфигурация исходного кода и проверка системы на возможность компиляции и последующего использования программы (configure скрипт).
- 2) Компиляция исходного кода при помощи утилиты **make** и компилятора **gcc**.
- 3) Тестирование скомпилированного кода (опционально).
- 4) Установка бинарных файлов и выполнение дополнительных настроек.

Конфигурация исходного кода и проверка системы выполняется при помощи запуска скрипта **configure** в каталоге с исходным кодом. Данный скрипт, как правило, создается разработчиком при помощи утилит **autoconf**, **automake** и **autoheader**. В его задачи входит проверка наличия необходимых системных библиотек, определение расположения программ, которые понадобятся в процессе сборки, определение архитектуры процессора и многое другое. В случае если скрипт **configure** не содержится в пакете, его можно получить, выполнив команду **autoreconf**²⁷ в каталоге с исходным кодом. Скрипт **configure** может иметь доста-

²⁶ **Макрогенерация** означает копирование входного символьного потока в выходной с подстановкой макросов по мере их появления. **Макросы** могут быть встроенными или определенными пользователями, и принимать произвольное число аргументов. Имеется множество встроенных функций для включения файлов, запуска внешних команд, выполнения целочисленной арифметики, манипуляции строками.

²⁷ Данная команда при необходимости выполняет команды **aclocal**, **autoconf**, **autoheader** и **automake** и создает **configure** скрипт.

точно большое количество опций, которые определяют настройки конечной программы. К таким опциям относятся префиксы расположения файлов программы, параметры компилятора и компоновщика, указания расположения заголовочных файлов и библиотек, определение ключевых возможностей программы. После завершения выполнения скрипта **configure** в сборочном каталоге будут присутствовать следующие файлы:

- **config.log**, содержащий выходную информацию скрипта **configure**;
- **config.h**, содержащий директивы препроцессора используемые при компиляции исходного кода;
- **config.status**, использующийся для формирования файлов **Makefile** и **config.h**;
- **config.cache**, использующийся для кэширования результатов при повторных выполнениях скрипта **configure**;
- **stamp.h**, использующийся для определения того, был ли изменен файл **config.h**;
- **Makefile**, содержащий инструкции по сборке исходного кода.

В случае, если система по каким-то причинам не удовлетворяет требованиям для установки данной программы, соответствующие сообщения об ошибках будут выданы пользователю.

После успешного выполнения скрипта **configure** необходимо выполнить сборку программы, используя утилиту **make**. Утилита **make** предназначена для автоматизированной сборки программ из исходного кода. В своей работе данная утилита использует собственный конфигурационный файл (make-файл), содержащий *правила* для сборки *целевых* объектов. Он может иметь названия **makefile**, **Makefile**, **GNUmakefile**. В этом случае программа **make** считывает его автоматически. Программа **make** вызывается следующим образом:

```
make [опции] [целевые_объекты] [макроопределения]
```

Опции, и целевые объекты и макроопределения задаются в любом порядке. Используется последнее присвоение переменной. Макроопределения указываются следующим образом:

```
имя=текст
```

```
или
```

```
имя:=текст
```

Правила в make-файле интерпретируются построчно. Если правило занимает больше одной строки, то следует использовать символ \. Make-файл может содержать следующие строки:

- *Пустые строки* (игнорируются при чтении).
- *Строки комментариев*(#). Игнорируется все, что следует за символом комментариев.
- *Строки зависимостей*. Содержат правила состоящие из целевых объектов

и предпосылок, и записываются в виде:

```
целевые_объекты :: предпосылка1 предпосылка2 ...
```

В первом случае (:) последующие команды выполняются, если предпосылки имеют более позднюю отметку времени, чем целевой объект. Во втором случае (::) допускается использовать одинаковые целевые объекты в нескольких строках. Если предпосылки не заданы, то последующие команды выполняются в любом случае. **Команды** указываются после предпосылок на следующей строке, после указания символа табуляции. Каждая команда выполняется в собственной командной оболочке и если какая-либо команда выполнится с ошибкой, происходит остановка процесса сборки и выход из программы **make**. Если перед командой стоит символ «-», то будут игнорироваться любые ошибки выполнения команды. Если перед командой указан символ «@», то утилита **make** не выводит на экран результаты своей работы.

Целевые объекты и предпосылки могут содержать символы подстановки, используемые в командном интерпретаторе.

Под **целевым объектом** принято понимать файл или действие, которое необходимо выполнить. В следующем листинге приведен простой пример make-файла:

```
foo.o: foo.c foo.h
    gcc -c foo.c
```

В данном листинге целевым объектом, который необходимо получить, является объектный файл **foo.o**. Команда **gcc -c foo.c** выполнится только в том случае, если существуют файлы **foo.c** и **foo.h**. В процессе обработки make-файла сначала происходит обновление файлов, указанных в предпосылках, а затем целевых объектов. **Предпосылка** представляет собой файл, использующийся для получения целевого объекта. Помимо правил, make-файл может содержать переменные, используемые затем в тех же правилах. Например, в следующем листинге приведен пример удаления созданных объектных файлов:

```
objects = main.o kbd.o command.o display.o \
    insert.o search.o files.o utils.o

edit : $(objects)
    cc -o edit $(objects)

.PHONY : clean
clean :
    rm edit $(objects)
```

В данном листинге переменной **objects** соответствует несколько объектных файлов. Далее следует правило получения целевого объекта **edit**. Затем идет описание псевдо целевого объекта, который представляет собой действие, т.е. команду удаления объектных файлов. Таким образом при вызове команды **make clean** будут выполнены команды удаления объектных файлов. Директива **.PHONY** явно указывает на то, что при вызове команды **make clean** следует выполнить команду **rm**, несмотря на то, что в make-файле может присутствовать целевой объект с именем **clean**.

Для того чтобы лучше понять зависимости между целными объектами и предпосылками, программа **make** использует *суффиксы*, указанные в make-файле. Суффиксы определяют какие файлы могут являться предпосылками для других файлов. Общая запись суффиксов имеет следующий вид:

`.суффикс1.суффикс2:`

В данном примере файлы, содержащие первый суффикс (`.суффикс1`) могут быть предпосылками для файлов, содержащих второй суффикс (`.суффикс2`). Запись `.с.о` обозначает, что файлы с расширением `.с` являются предпосылками для файлов с расширением `.о`.

Более совершенный механизм определения неявно указанных правил является использование *образцов*. Образец используется для сопоставления целевых объектов и предпосылок. Правила образцов содержат символ `%` в названии целевого объекта. В следующем листинге приведен пример правила, содержащего образцы:

```
%о : %.с
$(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@
```

Данная запись определяет правило, которое создаст любой файл с расширением `.о` из файла с расширением `.с`. В приведенном примере присутствуют две *внутренние переменные* (`$<` и `$@`), которые автоматически устанавливаются в зависимости от целевого объекта и его предпосылок. Переменная `$<` содержит имя первой предпосылки в правиле. Переменная `$@` содержит имя целевого объекта.

В следующем листинге частично приведен make-файл утилиты **tar** с указанием номеров строк:

```
1 SHELL = /bin/sh
2 #### Start of system configuration section. ####
3 srcdir = .
4 CC = gcc -O
5 YACC = bison -y
6 INSTALL = /usr/local/bin/install -c
7 INSTALLDATA = /usr/local/bin/install -c -m 644
8 CFLAGS = $(CDEBUG) -I. -I$(srcdir) $(DEFS) \
9         -DDEF_AR_FILE=\"$(DEF_AR_FILE)\" \
10        -DDEFBLOCKING=$(DEFBLOCKING)
11 LIBS =
12 LDFLAGS = -g
13 prefix = /usr/local
14 bindir = $(prefix)/bin
15 #### End of system configuration section. ####
16 SRC1 = tar.c create.c extract.c buffer.c \
17        getoldopt.c update.c gnu.c mangle.c
18 SRC2 = version.c list.c names.c diffarch.c \
19        port.c wildmat.c getopt.c
20 SRC3 = getopt1.c regex.c getdate.y
21 SRCS = $(SRC1) $(SRC2) $(SRC3)
22 OBJ1 = tar.o create.o extract.o buffer.o \
23        getoldopt.o update.o gnu.o mangle.o
24 OBJ2 = version.o list.o names.o diffarch.o \
```

```

25         port.o wildmat.o getopt.o
26 OBJ3 =  getopt1.o regex.o getdate.o $(RTAPELIB)
27 OBJS =  $(OBJ1) $(OBJ2) $(OBJ3)
28 .PHONY: all
29 all:    tar rmt tar.info
30 .PHONY: tar
31 tar:    $(OBJS)
32         $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LIBS)
33 rmt:    rmt.c
34         $(CC) $(CFLAGS) $(LDFLAGS) -o $@ rmt.c
35 tar.info: tar.texinfo
36         makeinfo tar.texinfo
37 .PHONY: install
38 install: all
39         $(INSTALL) tar $(bindir)/$(binprefix)tar
40         -test ! -f rmt || $(INSTALL) rmt /etc/rmt
41         $(INSTALLDATA) $(srcdir)/tar.info* $(infodir)
42         $(OBJS): tar.h port.h testpad.h
43         regex.o buffer.o tar.o: regex.h
44 .PHONY: clean
45         rm -f *.o tar rmt testpad testpad.h core

```

В 1-ой строке задается внутренняя переменная **SHELL**, определяющая командный интерпретатор. Далее следует описание общих переменных (строки 3-14), используемых в `make`-файле, таких как компилятор (**CC**), опции компилятора (**CFLAGS**), линковщика (**LDFLAGS**). Затем выполняется определение переменных, используемых в правилах (строки 16-27). В частности, в переменных **SRC1**, **SRC2**, **SRC3** перечисляются файлы с исходным кодом, а в переменных **OBJ1**, **OBJ2**, **OBJ3** – объектные файлы. После объявления переменных следуют шесть правил. Некоторые из них содержат имя специального целевого объекта **.PHONY**, что означает их выполнение даже в случае существования файлов с такими же именами.

Первое правило выполняет действие **all**, а именно создание файлов (предпосылок) **tar**, **rmt** и **tar.info**. Создание предпосылок выполняется в соответствующих правилах (строки 31-36). Пятое правило (строки 38-43) выполняет установку созданных целевых объектов. Шестое правило выполняет очистку сборочного каталога в случае ввода команды **make clean**.

Нетрудно заметить, что процедуры сборки и установки утилиты **tar** описаны в `make`-файле и выполняются при указании соответствующих целевых объектов. Для сборки программы, находясь в каталоге с исходным кодом, достаточно ввести команду

```
# make
```

Команда **make** считывает первое правило (целевой объект **all**) и выполняет сборку утилиты **tar**, **rmt**, и файла **tar.info**. Для перемещения созданных файлов в нужное место файловой системы, а также выполнения дополнительных настроек, требуется ввести команду

```
# make install
```

В результате выполнения данной команды произойдет выполнение команд,

указанных для целевого объекта **install**. Следует отметить, что непосредственная сборка бинарных файлов выполняется компилятором. В случае ручной сборки исходного кода пользователь вызывает управляющую программу, которая называется **gcc**. Она интерпретирует аргументы командной строки, определяет и запускает для каждого входного файла свои компиляторы нужного языка, запускает, если необходимо, ассемблер и компоновщик. Компилятор каждого языка является отдельной программой, которая получает исходный текст и порождает вывод на языке ассемблера. В случае автоматизированной сборки, вызов программы **gcc** выполняется утилитой **make** в процессе обработки make-файла. В следующем листинге приведен пример кода, написанного на языке C и выводящего на экран запись **Hello, world!**:

```
#include <stdio.h>
int
main (void)
{
printf ("Hello, world!\n");
return 0;
}
```

Для сборки данной программы в самом простом случае необходимо выполнить команду:

```
# gcc -Wall hello.c -o hello
```

В данном случае компилятору указывается опция вывода всех предупреждающих сообщений, а также файл с исходным кодом (**hello.c**) и результирующий бинарный файл – **hello**. Точно таким же образом происходит вызов компилятора **gcc** в make-файлах. В более сложных случаях указываются дополнительные опции компилятора, такие как:

- **I** – используется для указания дополнительных заголовочных файлов. По умолчанию поиск заголовочных файлов осуществляется в каталогах **/usr/local/include** и **/usr/include**. Для перманентного добавления дополнительных каталогов, содержащих заголовочные файлы, используются переменные окружения **C_INCLUDE_PATH** и **CPLUS_INCLUDE_PATH** (в случае использования языка C++);
- **L** – используется для указания дополнительных библиотек на этапе компоновки. По умолчанию поиск библиотек выполняется в каталогах **/usr/local/lib** и **/lib**. Для перманентного добавления дополнительных каталогов, содержащих библиотеки, используется переменная окружения **LIBRARY_PATH**;
- **W<уровень>** – используется для детализации вывода предупреждающих сообщений;

Некоторые исполняемые файлы требуют наличия *общих библиотек (shared libraries)*. Общие библиотеки должны быть загружены в память с диска до того, как будет выполнен исполняемый файл. Данные библиотеки позволяют снизить размеры исполняемых файлов и уменьшить потери дискового пространства, так как

могут быть использованы несколькими файлами одновременно. По умолчанию, поиск общих библиотек выполняется в каталогах `/lib` и `/usr/local/lib`. Если требуемая общая библиотека находится в другом каталоге, то необходимо указать путь для компановщика, используя переменную окружения `LD_LIBRARY_PATH`.



Модуль 2. Сетевое администрирование

После завершения изучения данного модуля вы научитесь:

- осуществлять базовое администрирование сервиса хранения учетных данных openLDAP;
- управлять хранением файлов при помощи сервисов NFS и Samba;
- осуществлять базовое администрирование сервиса DNS и DHCP;
- осуществлять базовое администрирование веб-сервера Apache и прокси-сервера Squid;
- осуществлять настройку сервиса SSH для обслуживания удаленных соединений;
- понимать основы администрирования почтового сервера Postfix;
- понимать и выполнять основные операции администрирования общих сетевых сервисов, XINETD, FTP и NTP.

2.1. Централизованное хранилище данных. Каталоги LDAP

В ОС Linux существует несколько сервисов, позволяющих централизованно управлять учетными записями пользователей и другой сервисной информацией. К таким сервисам относятся **NIS** и **LDAP**.

Сервис NIS позволяет хранить следующую информацию по учетным записям пользователей:

- имена пользователей;
- пароли пользователей;
- группы пользователей;
- расположения домашних каталогов.

В настоящее время на смену сервису NIS пришел более защищенный и производительный сервис каталогов LDAP.

При выборе способа управления учетными записями пользователей следует учитывать ОС, с которой работает большинство пользователей предприятия. Если для хранения учетных данных используется сервис LDAP, пользователи могут работать как на ОС Unix/Linux, так и на ОС Windows. Более того, в ОС Linux существует возможность реализовать подобие домена Active Directory для аутентификации пользователей с использованием протокола **Kerberos**.

Если же для хранения учетных данных используется NIS, пользователи могут

работать только на ОС Unix/Linux. Существуют, конечно, средства, позволяющие установить клиенты NIS на ОС Windows, однако данное решение не является разумным с точки зрения безопасности.

Для реализации сервиса каталогов LDAP в ОС Linux используется ПО **openLDAP**. Данное ПО можно разделить на следующие компоненты:

- серверы: предоставляют службы LDAP;
- клиенты: оперируют данными LDAP;
- утилиты: поддерживают работоспособность сервера LDAP;
- библиотеки: предоставляют API для доступа к данным LDAP;

Серверная часть сервиса LDAP представлена демоном **slapd**, который предоставляет доступ к одному или нескольким каталогам данных. Серверная часть LDAP может хранить данные локально или предоставлять доступ к внешним источникам данных. LDAP предоставляет возможности аутентификации компьютеров и пользователей, поиску и изменению данных, таких, например, как группы и адресная книга,

Клиенты, например, инструменты командной строки для поиска данных **ldapsearch** и изменения данных **ldapmodify** – получают доступ к сервису LDAP через LDAP-протокол. Их функция заключается в формировании и передаче запросов к демону **slapd**, который выполняет необходимые операции над данными в каталогах. Обычно сначала клиент подключается к демону **slapd**, аутентифицируется, а затем осуществляет необходимые операции, посылая LDAP запросы (поиск, изменение, удаление и пр.). После получения ответа на посланный запрос клиент завершает процесс биндинга и отключается (рис. 2.1).

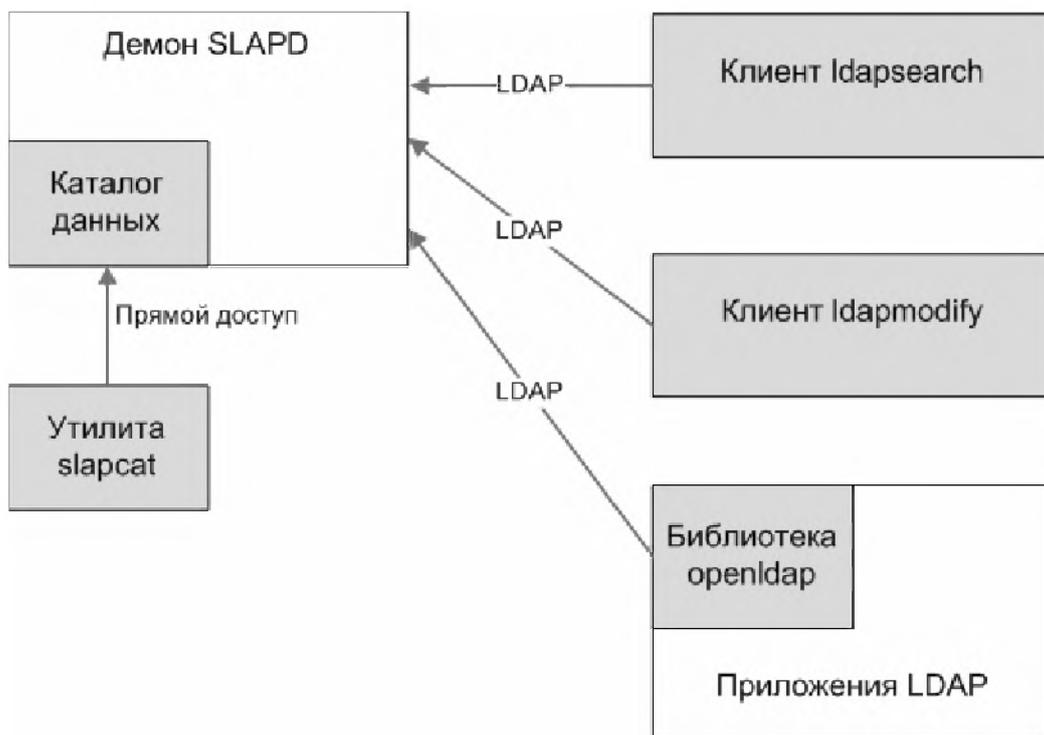


Рис. 2.1. Структура взаимодействия компонентов LDAP

Утилиты LDAP, в отличие от клиентов, не используют протокол LDAP для доступа к каталогам – они подключаются к серверу на более низком уровне и выполняют служебные операции над данными, например, создают новые каталоги. Данные утилиты используются в основном для сопровождения демона **slapd**.

В ОС Linux при инсталляции пакета **openLDAP** устанавливается несколько библиотек, которые используются LDAP-приложениями и позволяют сторонним разработчикам создавать ПО, взаимодействующее с каталогом по протоколу LDAP. Для развертывания сервиса LDAP в ОС Linux должны быть установлены пакеты **openLDAP** и **openLDAP-servers**. В общем случае процесс развертывания сервиса LDAP можно разделить на следующие этапы:

- установка бинарных пакетов **openLDAP** и **openLDAP-servers**;
- настройка сервера LDAP при помощи конфигурационного файла **slapd.conf**;
- проверка конфигурационного файла **slapd.conf** при помощи утилиты **slaptest**;
- создание каталога LDAP;
- настройка клиентов LDAP при помощи конфигурационного файла **ldap.conf**.

2.1.1. Настройка конфигурационных файлов LDAP

Все модули LDAP условно разделяются на хранилища данных LDAP (**backends**), в которых непосредственно хранятся записи каталога LDAP, и расширения LDAP (**overlay**), которые используются для добавления новых функциональных возможностей сервиса LDAP.

Основным конфигурационным файлом, который использует демон **slapd**, является файл **/etc/slapd.conf**. Данный файл содержит множество директив, которые разделяются на три группы: **Basics**, **Database Configuration** и **ACLs**.

В группе директив **Basics** выполняется:

- описание установленных схем (директива **include**);
- определение дополнительных системных параметров – уровня журналирования (директива **log level**) и аргументов запуска демона **slapd** (директива **argsfile**);
- определение используемых *модулей* LDAP (директивы **moduleload** и **modulepath**).

Под *схемой* в директиве **include** понимается служебная информация, описывающая разные классы объектов и типы атрибутов, которые поддерживает данный сервис LDAP. Кроме того, в данной группе описывается административная учетная запись пользователя **rootdn**.

В группе директив **Database Configuration** задаются:

- тип используемых баз данных каталогов (директива **database**);
- суффикс пространства имен (директива **suffix**);
- расположение файлов каталога (директива **directory**);
- индексируемые атрибуты (директива **index**).

Выбор типа базы данных каталога зависит от данных, которые необходимо хранить в этом каталоге. Если в нем хранятся главным образом данные пользователей или какие-то справочные данные, имеет смысл остановиться на иерархическом типе (**Hierarchical Database (HDB)**), он обеспечит наиболее быстрый поиск данных. Если в каталоге необходимо хранить различные данные, такие как документы pdf, видео- или аудиофайлы, то имеет смысл использовать файловую систему совместно с реляционной базой данных, которая будет использоваться для выполнения запросов к записям, ссылающимся на данные, находящиеся на файловой системе. В каталоге `/var/lib/ldap` должен присутствовать файл **DB_CONFIG**, образец которого содержится в каталоге `/etc/openldap`. Данный файл содержит опции, необходимые для настройки производительности базы данных каталога LDAP. Если демон **slapd** не обнаружит его в нужном каталоге, то будет выведено сообщение об ошибке.

Директива **suffix** определяет, для какой части каталога используется указанная база данных каталога. Данная директива определяет отличительное имя записей **Distinguish Name (DN)**.

В группе **ACLs** определяется, какие клиенты к каким данным каталога имеют доступ, а также уровень этого доступа.

После того, как конфигурационный файл **slapd.conf** будет создан, необходимо проверить его синтаксис, запустив команду **slaptest -v -f /etc/openldap/slapd.conf**. Если файл содержит ошибки, то в выводе будет указана строка, содержащая ошибку, и сама ошибка.

2.1.2. Создание каталога LDAP

Каждый объект в каталоге LDAP называется *записью*, а каждая запись обладает несколькими *атрибутами*, например, такими как имя (`name`) или расположение (`location`). Среди атрибутов есть обязательные и необязательные. Каждая запись идентифицируется своим отличительным именем **Distinguish Name (DN)**. Записи LDAP обычно состояются на основе использования атрибута **objectClass**. Классы объектов определяют атрибуты, которые может содержать запись. Каждому атрибуту назначается тип данных. При добавлении в каталог записи она должна удовлетворять требованиям *схемы* базы данных каталога.

Верхний уровень дерева класса объектов называется **top**; он определяет только то, что запись должна иметь атрибут **objectClass**. Некоторые схемы по умолчанию присутствуют в **openLDAP**. Администратор может добавлять дополнительные схемы и включать их в каталог, используя директиву **include**. Например, схема

dnszone.schema, входящая в состав пакета **bind-sdb**, позволяет хранить в каталоге LDAP зоны DNS. Большинство сетевых сервисов, например Samba, также имеют свои схемы, которые позволяют использовать каталог LDAP для хранения своих конфигурационных файлов.

Записи добавляются в каталог с использованием файла формата LDIF. Каждая запись в данном файле начинается с отличительного имени (DN), которое уникальным образом идентифицирует запись в каталоге. Запись состоит из нескольких атрибутов, записанных в отдельной строке. Каждый атрибут должен иметь значение, которое может быть представлено текстом в кодировке UTF-8, бинарными данными в кодировке base64, адресом URL или адресом файла (). Каждая запись в файле LDIF разделяется пробелом. Примеры двух записей показаны в следующем листинге.

```
# Организация: example.com
dn: dc=example, dc=com
dc: example
o: Softline
objectClass: top
objectClass: dcObject
objectClass: organization

# Организационная единица «маркетинг»: marketing
dn: ou=marketing, ou=employeedir, dc=example, dc=com
ou: marketing
objectClass: top
objectClass: organizationalUnit
```

Файл LDIF можно импортировать в каталог, используя команду **slapadd -v -l <имя_файла>.ldif**. Если синтаксис каждой записи корректен, то на терминал в процессе импорта будут выводиться сообщения о добавленных записях следующего вида:

```
added: "cn=Ivan Petrov,ou=accounting,ou=employeedir,dc=example,dc=com" (0000000a)
```

После добавления всех записей из файла LDIF необходимо установить соответствующие права для пользователя **ldap**, от имени которого запускается демон **slapd**, на все файлы базы данных каталога LDAP, находящихся в каталоге **/var/lib/ldap**. Запуск демона **slapd** осуществляется через стартовый скрипт **/etc/init.d/slapd** или при помощи команды **service**. Если запустить демон **slapd** при помощи стартового скрипта не удастся, то для определения причины данной проблемы можно запустить демон **slapd**, используя команду **slapd -d <config>**. Команда запускает демона **slapd** в отладочном режиме, при котором на терминал выводятся все сообщения отладки, относящиеся к указанному уровню. Уровень отладки указывается параметром для аргумента **-d**.

2.1.3. Подключение к серверу LDAP

Для того чтобы клиенты могли подключаться к серверу LDAP, в их системе должны быть установлены пакеты **openldap-clients** и **nss_ldap**. Данные пакеты

могут использовать как утилиты работы с каталогами LDAP, такие как **ldapsearch** (поиск записей) или **ldapadd** (добавление записей), так и конечные приложения, например, почтовый клиент **Evolution**.

Для того чтобы настроить ОС Linux в качестве клиента LDAP, необходимо добавить в файлы **/etc/openldap/ldap.conf** и **/etc/ldap.conf** IP-адрес сервера LDAP и корневую запись каталога, с которым будет осуществляться дальнейшая работа:

```
URI ldap://<IP_адрес_сервера_LDAP>/
BASE dc=example,dc=com
```

Если при подключении к серверу LDAP используется протокол TLS, то в данные файлы необходимо добавить путь к каталогу, в котором располагаются соответствующие сертификаты:

```
TLS_CACERTDIR /etc/openldap/cacerts
```

Если сервис LDAP используется для аутентификации пользователей, то необходимо отредактировать файл **/etc/nsswitch.conf** и добавить обращение системы к сервису LDAP в следующих записях:

```
passwd:      files          ldap
shadow:     files          ldap
group:      files          ldap
```

В данном случае при аутентификации пользователей, сначала будет просматриваться локальный файл **/etc/passwd**. Если пользователь имеет локальную учетную запись, то он будет аутентифицироваться на основе файла **/etc/passwd**. Если в файле **/etc/passwd** не содержится учетная запись пользователя, запрос аутентификации будет передан на сервер LDAP.

После того, как клиент LDAP будет настроен, необходимо протестировать подключение к серверу LDAP. Для проверки можно послать запрос для просмотра служебной информации о сервере LDAP, содержащейся записи **root DSE**:

```
# ldapsearch -x -W -D 'cn=admin,dc=example,dc=com' -b "" -s base
'(objectclass=*)' +
Enter LDAP Password: <P@ssw0rd>
dn:
structuralObjectClass: OpenLDAPProotDSE
configContext: cn=config
namingContexts: dc=example,dc=com
supportedControl: 1.3.6.1.4.1.4203.1.9.1.1
supportedControl: 2.16.840.1.113730.3.4.18
supportedExtension: 1.3.6.1.4.1.4203.1.11.1
supportedExtension: 1.3.6.1.4.1.4203.1.11.3
```

Для настройки аутентификации с использованием сервиса LDAP в ОС Linux существует графическая утилита **Red Hat Authentication Configuration tool**, которая запускается при помощи команды **system-config-authentication** (утилита имеет консольный вариант называющийся **authconfig-tui**). В ее настройках необходимо указать IP-адрес сервера LDAP и корневую запись каталога (**Base DN**), относительно которой будет выполняться поиск учетных записей пользователей.

2.2. Организация общего доступа к файлам (NFS, SMB)

В среде ОС Linux основным средством организации общей работы с данными является сетевая файловая система **NFS** (Network File System). Она позволяет пользователям совместно работать с данными, расположенными на удаленных хостах. В настоящее время NFS используется только для совместного использования файлов в средах Unix/Linux. Если в организации присутствуют Windows-клиенты, то для обслуживания файлов, которые они будут использовать, в ОС Linux используется сервис Samba. По умолчанию в ОС Linux используется протокол NFS версии 3, однако существует и версия 4.

2.2.1. Организация общего доступа на основе NFS

Взаимодействие хостов по протоколу NFS основано на архитектуре «клиент-сервер». Сервер NFS экспортирует указанные локальные файловые системы для клиента. Клиент, в свою очередь, монтирует данные файловые системы в точки монтирования. После того как файловая система будет смонтирована клиентом по NFS, все операции ввода-вывода к данной файловой системе адресуются серверу NFS, клиент же работает с данной файловой системой, как будто она является локальной для него. Контроль доступа к каталогам по протоколу NFS осуществляется по IP-адресам клиентов. В связи с этим сервис NFS, ничего не знает о пользователях, которые подключаются к нему. Данные, находящиеся на сервере NFS получают соответствующие права, идентификаторы пользователя (UID) и группы (GID) в процессе монтирования. Если клиент использует различные идентификаторы пользователей и групп, то владелец и группа файлов, доступных по NFS, будут меняться. Например, если владельцем файла является пользователь с идентификатором ID 500 на сервере NFS, то данный файл экспортируется клиентом с тем же идентификатором ID 500. Если идентификатор ID 500 принадлежит пользователю **user1** на сервере NFS, а на клиенте этот же идентификатор принадлежит пользователю **user2**, то пользователь **user2** также будет иметь доступ к данному файлу. Поэтому при использовании сервиса NFS крайне важно, чтобы каждый пользователь имел в сети уникальный идентификатор. Этого можно достичь, используя для хранения учетных записей пользователей сервисы NIS или LDAP.

В NFS версии 3 процесс монтирования файловой системы отличается от процесса последующего доступа к файлам. В данных операциях используются совершенно разные протоколы, и запросы обслуживаются разными демонами: в первом случае – это **mountd**, во втором – **nfsd**. На самом деле эти демоны называются соответственно **rpc.mountd** и **rpc.nfsd** – имена служат напоминанием о том, что их работа основана на механизме **RPC (Remote Procedure Call)**. Для их выполнения требуется демон **portmap**. На сервере NFS оба демона – **mountd** и **nfsd** – должны

запускаться на этапе начальной загрузки системы и выполняться, пока система не будет выключена. В ОС Linux существует единый init-скрипт `/etc/init.d/nfs`, который запускает необходимые демоны NFS автоматически, при условии, что экспортируемые файловые системы настроены должным образом.

Демоны **mountd** и **nfsd** используют общую базу данных, содержащую информацию о правах доступа, где указано, какие файловые системы следует экспортировать и какие клиенты могут их монтировать. Рабочая копия этой базы данных хранится в файле `/usr/lib/nfs/xtab`. Поскольку файл базы данных представлен в бинарном формате, для добавления и редактирования записей существует вспомогательная утилита **exportfs**.

В ОС Linux имеется файл `/etc/exports`, содержащий список экспортируемых каталогов в текстовом виде. По умолчанию все файловые системы, указанные в этом файле, экспортируются на этапе начальной загрузки. С помощью команды **exportfs -a** можно самостоятельно выполнить экспорт после внесения изменений в файл `/etc/exports`. В командной строке утилиты **exportfs** можно указать имя клиента, путь к нужному каталогу и различные опции.

NFS работает не на физическом, а на логическом уровне файловой системы. Экспортировать можно любой каталог; не обязательно, чтобы он был точкой монтирования или корневым каталогом физической файловой системы. Тем не менее из соображений безопасности NFS определяет границы между файловыми системами, относящимися к различным устройствам, и требует, чтобы каждая такая файловая система экспортировалась отдельно.

Обычно клиентам разрешается монтировать подкаталоги экспортированного каталога. Например, если сервер экспортирует каталог `/softline/users`, то клиент может смонтировать подкаталог `/softline/users/andrew`, а остальную часть каталога **users** – проигнорировать:

```
/home/andrew          client1(rw,no_root_squash) client2(rw)
/usr/share/man        *.linux.lab(ro)
```

Каждая файловая система отделена от списка клиентов пробелом, а сразу после имени клиента в круглых скобках стоит список связанных с ним опций, разделенных запятыми. Длинные строки разрешается разбивать на части с помощью обратной косой черты.

В табл. 2.1 и 2.2 указаны возможные параметры для файла `/etc/exports`.

Таблица 2.1. Варианты указания клиентов NFS

Тип	Синтаксис	Назначение
Имя хоста	имя_хоста	Используется для указания отдельных хостов
Символы подстановки	* и ?	Используются для подстановки в доменных именах хостов
IP-адреса	<IP-адрес>/<маска>	Используются для указания IP-адресов хостов и подсетей, например, 192.168.1.2/25

Таблица 2.2. Основные опции экспорта файловых систем через NFS

Опция	Описание
ro и rw	Доступ только для чтения и доступ для чтения/записи (используется по умолчанию)
rw=	Доступ для чтения/записи только для указанных клиентов
no_root_squash	Разрешает обычный доступ от имени пользователя root
root_squash	Подмена значений UID и GID, равных 0, значениями, указанными в опциях anonuid и anongid; это установка по умолчанию
anonuid=NNN	Значение UID, которое нужно использовать для запросов, поступающих от пользователя root
anongid=NNN	Значение GID, которое нужно использовать для запросов, поступающих от пользователя root
secure	Требует, чтобы запрос на удаленный доступ поступал с привилегированного порта
insecure	Разрешает удаленный доступ с любого порта
async	Заставляет сервер отвечать на запросы на запись, прежде чем выполнять сами операции записи на диск
sync	Заставляет сервер выполнять сами операции записи на диск, прежде чем отвечать на запросы записи
noaccess	Предотвращает доступ к каталогу и его подкаталогам (используется при экспорте вложенных каталогов)
hide и nohide	Не показывает/показывает файловые системы, смонтированные внутри экспортированных каталогов

Сервер NFS в Linux позволяет экспортировать родительский каталог с одними опциями, а подкаталоги – с другими. Имеется также опция **noaccess**, позволяющая отменить экспорт подкаталогов, к которым не нужно предоставлять удаленный доступ.

Например, в конфигурации

```
/home *.linux.lab(rw)
/home/teacher (noaccess)
```

хостам домена linux.lab разрешен доступ ко всем каталогам файловой системы **/home**, кроме **/home/teacher**. Отсутствие имени клиента во второй строке означает, что установка относится ко всем хостам.

После внесения изменений в файл **/etc/exports** необходимо выполнить команду **exportfs -a**, для того чтобы изменения вступили в силу.

Демон **nfsd** принимает числовой аргумент, определяющий, сколько экземпляров самого себя нужно породить посредством системного вызова **fork**. По умолчанию используется восемь потоков **nfsd**. На загруженном сервере данный показатель должен быть в диапазоне от 12 до 20. Список параметров запуска демона **nfsd** можно указать в файле **/etc/sysconfig/nfs** или же исправить init-скрипт **/etc/init.d/nfs**. Кроме того, в файле **/etc/sysconfig/nfs** можно указать статические порты,

которые будут использовать демоны NFS, что может пригодиться при настройке правил для сервиса NFS на межсетевом экране.

Настраивать сервис NFS быстрее и проще всего путем редактирования конфигурационного файла `/etc/exports`. Однако в ОС Linux для такой задачи существует и графическая утилита, **NFS Server Configuration tool**, запустить которую можно при помощи команды **system-config-nfs**.

Процесс монтирования сетевых и локальных файловых систем во многом схож. Для монтирования сетевой файловой системы, доступной по NFS, используется следующая команда:

```
mount -o <опции_монтирования> имя_хоста:каталог <точка_монтирования>
```

Команда **mount** распознает запись вида **имя_хоста:каталог** как путь к каталогу, расположенному на указанном хосте. Этому каталогу будет поставлен в соответствие каталог локальной файловой системы, указанный в аргументе **<точка_монтирования>**. После завершения монтирования доступ к сетевой файловой системе осуществляется традиционными средствами.

Параметры, указанные после опции **-o**, говорят о том, что файловая система монтируется в режиме чтения/записи, файловые операции разрешается прерывать, а повторные попытки монтирования должны происходить в фоновом режиме. Основные опции монтирования приведены в табл. 2.3.

Таблица 2.3. Основные опции монтирования файловых систем через NFS

Параметр	Описание
rw и ro	Монтирование файловой системы для чтения/записи и только для чтения
bg	Если смонтировать файловую систему не удастся, следует перевести операцию в фоновый режим и продолжить обработку других запросов на монтирование
hard	Если сервер отключился, операции, которые пытаются получить к нему доступ, блокируются до тех пор, пока сервер не включится вновь
soft	Если сервер отключился, операции, которые пытаются получить к нему доступ, завершаются выдачей сообщения об ошибке. Этот флаг полезно устанавливать для того, чтобы предотвратить зависание процессов в случае неудачного монтирования не очень важных файловых систем
intr	Позволяет прерывать с клавиатуры заблокированные операции (будут выдаваться сообщения об ошибке)
nointr	Не позволяет прерывать с клавиатуры заблокированные операции
retrans=N	Указывает, сколько раз нужно повторить запрос, прежде чем будет выдано сообщение об ошибке (для файловых систем, смонтированных с опцией soft)
timeo=N	Задаёт интервал тайм-аута для запросов (в десятых долях секунды)

Параметр	Описание
rsize=N	Задаёт размер буфера чтения равный N байт
wsizе=N	Задаёт размер буфера записи равный N байт
tcp	Использовать TCP в качестве транспортного протокола: по умолчанию используется протокол UDP
async	Заставляет сервер отвечать на запросы на запись, прежде чем будет производиться запись данных на диск

Для того чтобы файловую систему NFS можно было смонтировать, ее сначала необходимо корректно экспортировать. Для проверки экспортированных файловых систем на клиенте присутствует команда **showmount**, отображающая экспортированные на сервере файловые системы.

2.2.2. Организация общего доступа на основе Samba

В ОС Linux имеются средства, предназначенные для организации общего доступа к файловым системам для Windows-клиентов. Сервис Samba запускается как пользовательский процесс. Он устанавливает соединения с сокетами, через которые посылаются CIFS-запросы, и ждет клиентских запросов на доступ к общим ресурсам. Как только запрос поступил и был аутентифицирован, демон **smbd** создает новый процесс **smbd**, который работает от имени пользователя, пославшего запрос. Таким образом, все права доступа к файлам (включая групповые права) остаются ненарушенными.

Сервис Samba можно рассматривать как серверную часть, в случае предоставления общего доступа к данным для Windows-клиентов, и как клиентскую часть, в случае доступа к данным, расположенным на серверах Windows.

Для того чтобы в ОС Linux организовать общий доступ на основе сервиса Samba необходимо установить следующие пакеты:

- **samba**, содержащий серверную часть Samba;
- **samba-client**, содержащий утилиты, при помощи которых можно подключаться к общим каталогам ОС Windows;
- **system-config-samba**, который содержит графическую утилиту управления сервисом Samba, **Red Hat Samba Server Configuration utility**;
- **samba-common**, который содержит общие конфигурационные файлы Samba.

Конфигурирование сервиса Samba

Конфигурационные файлы сервиса Samba располагаются в каталоге **/etc/samba**. Основным конфигурационным файлом серверной части Samba является файл **/etc/samba/smb.conf**. В данном файле определяются следующие группы настроек:

- **Глобальные (global).** В данной группе настроек определяется рабочая группа (**workgroup**), по смыслу аналогичная рабочим группам, используемым в ОС Windows; NetBios – имя данного сервера (**netbios name**), которое используется для того, чтобы клиенты Windows могли просматривать общие папки сетевого окружения; сетевые интерфейсы (**interfaces**), используемые для обработки клиентских соединений; список клиентских IP-сетей (**hosts allow**), которым разрешено подключаться к данному серверу. Кроме того, в данной группе настроек присутствует директива **security**, которая определяет, будет ли данный сервер использоваться в качестве локального хранилища учетных записей пользователей. По умолчанию значением данной директивы является значение **user**. Если же в директиве указано значение **ads**, то сервер Samba будет использоваться как сервер, находящийся в домене Active Directory. В директиве **passdb backend** содержится база данных, используемая для хранения паролей. По умолчанию в данной директиве указан параметр **tdbsam**, что соответствует базе данных **trivial database**. Для хранения паролей можно также использовать файл **smbpasswd**, либо каталог LDAP.
- **Настройки общих ресурсов (homes, printers).** Данные настройки используются для организации общего доступа пользователей к своим домашним каталогам, а также к общим сетевым принтерам. Каждый пользователь имеет доступ только к своему домашнему каталогу, поскольку по умолчанию используется директива **browseable=no**.

Для того чтобы запустить сервис Samba в работу необходимо запустить два демона: **smbd** и **nmbd**, последний из которых используется для работы с именами NetBios. Демоны считывают свои настройки из файла **smb.conf**. Помимо данных демонов в состав сервиса Samba входят следующие утилиты:

- **/usr/bin/smbclient**, используемая для подключения к общим ресурсам SMB;
- **/sbin/mount.cifs**, которая используется для монтирования общих ресурсов SMB;
- **/sbin/umount.cifs**, используемая для размонтирования общих ресурсов SMB;
- **/etc/init.d/smb** – сценарий запуска сервисов Samba;
- **/usr/bin/smbprint** – сценарий, использующийся для печати данных на принтер, доступный через SMB хост;
- **/usr/bin/smbstatus**, которая отображает текущие соединения клиентов с демоном **smbd**.

Создание общих ресурсов SMB

Сервис Samba по умолчанию использует собственную базу данных пользователей. Для добавления нового пользователя в эту базу необходимо выполнить команду:

```
smbpasswd -a <имя_пользователя>
```

В параметре **<имя_пользователя>** необходимо указать имя пользователя, который уже присутствует в системе в файле **/etc/passwd**. Данная команда записывает зашифрованный пароль пользователя в файл **/etc/samba/smbpasswd**. Использование шифрования паролей задается в директиве **encrypted passwords = yes** глобальных настроек файла **smb.conf**.

Если к данному серверу Samba планируется подключать пользователей Windows, в файле **/etc/samba/smbusers** необходимо задать соответствие между именами пользователей Samba и пользователей Windows в следующем формате:

```
username = Windows_user1
```

Для соотношения нескольких пользователей Windows одному пользователю Linux необходимо перечислить пользователей Windows через пробел в директиве **username**:

```
[data]
comment=Private share for user1 and user2
path=/shares/data
read only = no
valid users = user1 user2
browseable = no
```

В данном листинге определен общий ресурс **data**, который будет доступен для чтения и записи только двум пользователям – **user1** и **user2**.

После того, как конфигурационный файл **/etc/samba/smb.conf** будет отредактирован, его необходимо проверить на наличие ошибок при помощи команды **testparm**.

По умолчанию сервисы **smbd**, **nmbd** и **mount.cifs** записывают системные события в файл **/var/log/messages**. Помимо журналирования системных данных демонов существует возможность журналировать информацию о подключениях клиентов. Для этого в файле **smb.conf** должна присутствовать следующая директива:

```
log file = /var/log/samba/%m.log
```

Согласно данной директиве в файл **/var/log/samba/<имя_клиента>.log** будет журналироваться информация по каждому клиенту, который подключается к демону **smbd**.

Подключение к ресурсам SMB и их монтирование

Утилита **smbclient** предоставляет доступ к ресурсам SMB. Для того чтобы подключиться к ресурсу SMB необходимо знать его название. Если название ресурса неизвестно, то можно подключиться к серверу Samba и просмотреть список всех доступных ресурсов, используя следующую команду:

```
smbclient -L <имя_сервера> -U <имя_пользователя>
```

Для подключения к конкретному ресурсу SMB необходимо использовать следующую команду:

```
smbclient //<имя_сервера>/<название_ресурса> -U <имя_пользователя>
```

При удачном подключении в терминале отобразится приглашение клиента **smbclient** вида **smb: \>**. После того как подключение было успешно установлено, можно выполнять различные команды над файлами, находящимися на данном ресурсе.

Для того чтобы смонтировать ресурс SMB необходимо использовать команду **mount**:

```
mount -t cifs //<имя_сервера>/<название_ресурса> /<точка_монтирования> -o  
username=<имя_пользователя>
```

Для размонтирования ресурса SMB, используется штатная команда **umount**. Монтирование ресурсов SMB может быть настроено при помощи сервиса автоматического монтирования **autofs**. Например, для монтирования ресурса **/misc/** **<точка_монтирования>** в файле **/etc/auto.master** должна присутствовать следующая запись:

```
/misc /etc/auto.misc
```

После того как данная запись будет добавлена, необходимо указать параметры ресурса SMB в файле **/etc/auto.misc**:

```
mount_dir -fstype=cifs,credentials=/etc/smbcreds \ ://<имя_сервера>/<название_ресурса>
```

Как видно из данного примера, при монтировании ресурсов SMB параметры учетной записи пользователя можно указывать в отдельном файле, например, **/etc/smbcreds**.

2.3. Система разрешения имен (DNS). Автоматизация получения сетевых параметров (DHCP)

2.3.1. Сервис DNS

Сервисом DNS, присутствующим в ОС Linux, является ПО **Berkeley Internet Name Domain (BIND)**. В данном разделе мы рассмотрим конфигурирование данного ПО с целью организации DNS-сервера.

Для настройки сервера DNS в ОС Linux необходимо установить пакеты **bind**, **bind-utils** и ответствующие им зависимости. Проще всего устанавливать сервер DNS, используя менеджер пакетов **yum**. В качестве основного конфигурационного файла ПО BIND использует файл **/etc/named.conf**, а для управления сервером DNS при помощи утилиты **rndc** – файл **/etc/rndc.conf**. База данных имен DNS находится в каталоге **/var/named** и содержит файлы описания прямых и обратных зон DNS. Данные файлы имеют текстовый формат, поэтому их можно редактировать в текстовом редакторе или же использовать специальную графическую программу, запускаемую командой **system-config-bind**.

В каталоге **/usr/share/doc/bind-<версия>/arm** можно ознакомиться с руко-

водством «**BIND 9 Administrator Reference Manual**». Руководство содержит информацию, начиная с фундаментальных понятий DNS и требований к ПО BIND и до настройки и защиты сервиса DNS.

Конфигурационный файл сервиса DNS – **named.conf** является основным конфигурационным файлом. Владельцем данного файла должен быть пользователь **named**, так как демон **named** запускается именно из-под этой учетной записи. Код доступа данного файла должен позволять просматривать и изменять его только пользователям **named** и **root**. В конфигурационном файле демона **named**, **named.conf**, задается роль (главный, подчиненный или ограниченный) сервера и определяется способ, которым он должен получать копию данных для каждой из обслуживаемых им зон. Здесь же приводятся всевозможные параметры – глобальные, связанные с работой самого демона, и локальные, применяемые только к данным DNS.

Конфигурационный файл состоит из набора **инструкций**, каждая из которых оканчивается точкой с запятой. Лексемы разделяются пробельными символами, к которым относится и символ новой строки. Иногда для группировки лексем применяются фигурные скобки. Формат файла довольно строгий: достаточно одной пропущенной точки с запятой, чтобы все перестало работать.

В ПО BIND имеются специальные средства проверки синтаксиса конфигурационного файла (**named-checkconf**) и зонных файлов (**named-checkzone**). Эти утилиты ищут не только синтаксические ошибки, но и очевидные пропуски. Например, утилита **named-checkzone** выдает предупреждение, если в файл не включена директива **\$TTL** (время жизни записей зоны). Комментарии допускаются везде, где могут стоять пробелы.

Каждая инструкция начинается с ключевого слова, определяющего ее тип. Может присутствовать несколько инструкций одного типа, за исключением **options** и **logging**. Отдельные инструкции, а также их части могут отсутствовать; в этом случае будут приняты установки по умолчанию.

Инструкция include используется для включения дополнительных файлов в состав файла **named.conf**. Данные файлы могут иметь более жесткие права доступа, необходимые для более надежной защиты данных. Параметр **<имя_файла>** должен включать полный путь к включаемому файлу:

```
include "<имя_файла>"
```

Если указан относительный путь, он добавляется к имени каталога, заданному в параметре **directory**. Очень часто инструкцию **include** применяют для подключения файлов, содержащих ключи шифрования. Эти данные должны быть доступны только демону **named**. Чтобы не запрещать доступ ко всему файлу **named.conf**, ключи хранят в отдельных файлах, которые может читать лишь демон **named**.

Инструкция options задает глобальные параметры конфигурации, часть которых впоследствии может быть переопределена для конкретных зон или серверов. Общий формат инструкции следующий:

```
options {  
    параметр;  
    параметр;
```

```
};
```

В версии BIND 9 существует около 100 параметров.

Инструкция `acl` содержит списки доступа в следующем формате:

```
acl имя_списка {  
    список_доступа  
};
```

Заданное имя списка можно указывать везде, где требуется список соответствия адресов.

Инструкция **`acl`** должна быть самой первой в файле **`named.conf`**, так как он считывается один раз, и списки управления доступом должны быть определены до того, как на них встретится ссылка.

В качестве параметра **`список_доступа`** могут использоваться IP-адреса хостов и подсетей в битовом формате записи сетевой маски, стандартные списки доступа; или диапазон IP-адресов. Например, в следующем списке доступа разрешены подсети 172.16.0.1 и 192.168.1.0 и запрещен IP-адрес 192.168.1.5:

```
{ 172.16.0.1/24; 192.168.1.0/24 };  
{ ! 192.168.1.5 };
```

По умолчанию существуют следующие стандартные списки доступа:

- `any` – все хосты;
- `localnets` – все хосты локальной сети;
- `localhost` – локальный хост;
- `none` – ни один из хостов.

Сети, входящие в группу **`localnets`**, определяются адресами сетевых интерфейсов хоста с учетом сетевых масок.

Инструкция `key` определяет ключ шифрования, используемый для аутентификации на сервере с использованием механизма **`TSIG`**. Для создания ключа нужно указать алгоритм шифрования и секретный ключ, который представлен в виде строки, закодированной в формате `base64`:

```
key идентификатор_ключа {  
    algorithm строка;  
    secret строка;  
};
```

Идентификатор ключа должен быть определен с помощью инструкции **`key`** в файле **`named.conf`** до того, как встретится ссылка на ключ. Чтобы связать ключ с конкретным сервером, необходимо включить идентификатор ключа в список **`keys`** соответствующей инструкции **`server`**. Ключ используется для проверки запросов, поступающих от сервера, а также для подписи ответов на эти запросы.

Инструкция `trusted-keys` является частью механизма аутентификации **`DNSSEC`**, описанного в документе RFC-2535. Каждая запись состоит из пяти компонентов, определяющих имя домена, флаги, протокол, алгоритм шифрования и ключ. Все это необходимо для безопасного взаимодействия с сервером имен домена. Формат инструкции следующий:

```
trusted-keys {
домен флаги протокол алгоритм ключ;
домен флаги протокол алгоритм ключ;
...
};
```

Каждая строка представляет ключ конкретного домена. Атрибуты **флаги**, **протокол** и **алгоритм** являются неотрицательными целыми числами. Атрибут **ключ** – это строка, закодированная в формате base64.

Инструкция **trusted-keys** используется в тех случаях, когда зона имеет цифровую подпись, а ее родительская зона – нет, поэтому нельзя быть уверенным в том, что открытый ключ, получаемый от DNS-сервера, действительно надежный.

Инструкция server. Демон **named** способен общаться с серверами, которые не используют последнюю версию пакета BIND или просто неправильно настроены. Инструкция **server** сообщает демону характеристики удаленных серверов.

```
server IP-адрес {
    bogus yes|no;                [no]
    provide-ixfr yes|no;         [yes]
    request-ixfr yes|no;        [yes]
    edns yes|no;                 [yes]
    transfers число;             [2]
    transfer-format one-answer|many-answers; [many-answers]
    keys { ключ; ключ; ... };
    transfer-source IP-адрес [порт]
    transfer-source-v6 IP-адрес^6 [порт]
};
```

С помощью инструкции **server** можно переопределять значения глобальных конфигурационных параметров, относящихся к серверам.

Параметры **transfers** и **transfer-format** ограничивают количество одновременных входящих зонных пересылок в секунду от удаленного сервера.

В списке **keys** задаются идентификаторы ключей, которые ранее были определены в инструкции **key** для использования в сигнатурах транзакций TSIG.

Записи **transfer-source** предоставляют адрес (IPv4 или IPv6) интерфейса, который нужно использовать в качестве исходного адреса (порта) запросов зонной пересылки.

Инструкция masters позволяет указать один или несколько главных серверов с помощью IP-адресов и ключей шифрования. Указанное имя затем можно использовать в параметре **masters** инструкций **zone**, чтобы не повторять IP-адреса и ключи. Данная инструкция имеет следующий синтаксис:

```
masters имя { IP-адрес [порт номер_порта] [key ключ]; ... };
```

Инструкции zone – одни из самых главных инструкций файла **named.conf**. Они сообщают демону **named** о зонах, для которых он авторитетен, и задают параметры управления каждой зоной. Точный формат инструкции **zone** зависит от роли, которую демон **named** должен играть в отношении этой зоны. Возможными типами зон являются **master**, **slave**, **hint**, **forward**, **stub** и **delegation-only**.

Ниже показан формат инструкции **zone** для зоны, в которой демон **named** является главным сервером имен:

```
zone "имя_домена" {
    type master;
    file "путь";
};
```

Доменное имя в спецификации зоны всегда дается в двойных кавычках.

Зонная база данных хранится на диске в текстовом файле, доступном для редактирования. Поскольку нет соглашения об именовании этого файла, в объявлении зоны должна присутствовать директива **file**. Зонный файл представляет собой набор записей о DNS ресурсах и имеет специальный формат, подробно описанный в руководстве «**BIND 9 Administrator Reference Manual**».

```
zone "localhost" { // Зона прямого преобразования localhost
    type master;
    file "fwd/localhost"; allow-update { none; };
zone "0.0.127.in-addr.arpa" { // Зона обратного преобразования localhost
    type master;
    file "rev/127.0.0";
        allow-update { none; };
};
```

Из приведенного листинга видно, что файл, содержащий прямую зону, находится в каталоге **fwd**, а файл, содержащий обратную зону, – в каталоге **rev**. В данном случае используются относительные пути данных каталогов. Соответствующий файл прямой зоны для данного примера указан в следующем листинге.

```
$TTL 30d
; localhost.
@           IN  SOA   localhost. postmaster.localhost. (
                                1998050801 ; порядковый номер
                                3600         ; период обновления
                                1800         ; интервал между попытками
обновления
                                604800      ; интервал устаревания
                                3600 )      ; минимальное время жизни
                NS  localhost.
                A   127.0.0.1
```

Инструкция **controls** определяет, каким образом команда **rndc** будет управлять работой демона **named**. Эта команда может запускать и останавливать демон, выводить отчет о его состоянии, переводить демон в режим отладки и т.д. Формат инструкции **controls** следующий:

```
controls {
    inet IP-адрес port номер_порта allow {
        список_соответствия_адресов } keys { список_ключей };
}
```

Если параметр **ports** опущен, то по умолчанию **rndc** будет использовать порт 953 для обмена данными с демоном **named**. В качестве значения параметра **IP-адрес** рекомендуется использовать локальный IP-адрес 127.0.0.1.

Инструкция **view** содержит список адресов, определяющий, кто из клиентов имеет доступ к данному представлению, а также ряд параметров, применимых ко всем зонам в представлении, и определения самих зон. DNS сервер BIND позво-

ляет настраивать способ представления данных зоны в зависимости от IP-адреса хоста, который выполняет запрос. Синтаксис инструкции таков:

```
view имя_представления {  
    match-clients { список_соответствия_адресов };  
    параметр_представления; ...  
    определение_зоны; ...  
};
```

В списке **match-clients** перечислены клиенты представления. Представления просматриваются по порядку, поэтому инструкции **view** с самыми большими ограничениями доступа должны идти впереди. Зоны в разных представлениях могут иметь одинаковые имена и принимать свои данные могут из разных файлов. Представления налагают ограничение на структуру файла **named.conf**: если они присутствуют, то все инструкции **zone** должны находиться в контексте представлений.

Для создания файла **named.conf** можно воспользоваться файлом **/usr/share/doc/bind-<версия>/sample/etc/named.conf**, который представлен как образец. Таким же образом можно поступить при создании файлов зон, образцы которых находятся в каталоге **/usr/share/doc/bind-<версия>/sample/var/named**.

Для управления сервером DNS в состав пакета **bind-utils** входит утилита **rndc**, которая посылает команды управления, подписанные цифровой подписью, на сервер DNS. В процессе работы данная утилита использует файл **/etc/rndc.conf**, в котором содержатся имя сервера DNS и имя ключа, который необходимо использовать для подписи команд. Для создания файла **rndc.conf** можно воспользоваться командой **rndc-confgen > /etc/rndc.conf**. После того, как в каталоге **/etc** файл **rndc.conf** будет создан, можно использовать утилиту **rndc** для управления сервером DNS.

Основным конфигурационным файлом клиента DNS является файл **/etc/resolv.conf**. Он должен содержать минимум две записи – **nameserver** и **search**. Первая из них определяет IP-адрес сервера DNS, а вторая определяет доменное имя, которое будет добавляться при формировании клиентских запросов, не содержащих полного доменного имени:

```
nameserver 192.168.137.2  
search linux.lab
```

Для того чтобы это стало возможным разрешения имен хостов через сервис DNS необходимо еще отредактировать файл **/etc/nsswitch.conf**, который определяет, какие сервисы необходимо опрашивать для получения данных о хостах. Данный файл должен содержать строку:

```
hosts: files,dns
```

В данном случае при разрешении имен хостов сначала будет просматриваться локальный файл **/etc/hosts**, а затем (если указанное имя не было найдено) будет выполняться запрос на DNS-сервер.

2.3.2. Сервис DHCP

DHCP (Dynamic Host Configuration Protocol) дает возможность клиенту получать сетевые и административные параметры с центрального сервера, отвечающего за их распространение.

Данные параметры клиент получает на некоторое определенное время. Протокол DHCP оперирует понятием «*аренды IP-адреса*». По истечении половины срока аренды клиент должен ее продлить. Сервер должен отслеживать адреса, предоставленные в аренду, и сохранять эту информацию при перезагрузке. Если сервера DHCP нет, сообщение может быть передано в другие подсети через специальный прокси-сервер, называемый *агентом ретрансляции*.

В ОС Linux сервер DHCP представлен ПО **ISC DHCP**, которое содержится в пакете **dhcp**, и использует в качестве своего основного конфигурационного файла **/etc/dhcpd.conf**.

Для конфигурирования DHCP-сервера **dhcpd** нужно отредактировать файл **dhcpd.conf.sample**, который находится в каталоге **/usr/share/doc/dhcp-<версия>**, и записать его под именем **/etc/dhcpd.conf**. Необходимо также создать пустой файл базы данных по арендуемым параметрам, назвав его **/var/lib/dhcpd/dhcpd.leases**. После этого необходимо убедиться в том, что демон **dhcpd** имеет право записи в этот файл. Для заполнения файла **dhcpd.conf** потребуется следующая информация:

- адреса подсетей, в которых демон **dhcpd** должен управлять IP-адресами, и диапазоны выделяемых адресов;
- начальный и максимальный сроки аренды в секундах;
- конфигурационные параметры клиентов BOOTP, если таковые имеются (им назначаются статические IP-адреса, также должны быть указаны их аппаратные MAC-адреса);
- все остальные параметры, которые сервер должен передавать DHCP-клиентам: сетевая маска, стандартный маршрут, домен DNS, адреса серверов имен и т.д.

На страницах руководства **man**, посвященных демону **dhcpd**, дан обзор процесса конфигурации. Точный синтаксис конфигурационного файла описан на man странице файла **dhcpd.conf**.

Демон **dhcpd** должен автоматически запускаться на этапе начальной загрузки системы. Для этого в ОС Linux имеется сценарий его автозапуска **/etc/init.d/dhcpd.conf**. Для задания дополнительных аргументов демону **dhcpd** в процессе начальной загрузки необходимо отредактировать запись **DHCPDARGS=** в файле **/etc/sysconfig/dhcpd**.

В типовом файле **dhcpd.conf** директива **ddns-update-style interim** указывает на то, что используется механизм динамического обновления DNS, при котором база DNS-имен обновляется после того, как сервер DHCP обновит IP-адрес. Директива **ignore client-updates** не позволяет пользователям изменять свои имена хостов. Далее в директиве **subnet** указывается пул IP-адресов, которые DHCP-

сервер предоставляет в аренду своим клиентам. Затем следует несколько директив, позволяющих клиенту определить сервер DNS, шлюз по умолчанию и маску подсети. В директиве **range** выделяется диапазон IP-адресов, которые можно присваивать клиентам. В директивах **default-lease-time** указывается время аренды IP-адреса (в секундах), а в директиве **max-lease-time** – и максимальное время аренды IP-адреса (в секундах). В конце файла **dhcpd.conf** указан сервер DNS, для которого на данном DHCP-сервере зарезервирован какой-то IP-адрес.

DHCP-клиент в ОС Linux не требует особого конфигурирования. Для его работы достаточно установить пакет **dhclient** и в конфигурационном файле сетевого интерфейса, например, в **/etc/sysconfig/network-scripts/ifcfg-eth0**, добавить следующие записи:

```
BOOTPROTO='dhcp'  
ONBOOT='yes'
```

Файлы с информацией о статусе каждого соединения клиента DHCP хранятся в каталоге **/var/lib/dhclient**. Имя файла соответствует имени описываемого интерфейса. Например, файл **dhclient-eth0.leases** будет содержать все сетевые параметры, которые демон **dhclient** закрепил за интерфейсом **eth0**.

2.4. Администрирование веб-сервера Apache

В ОС Linux основным веб-сервером является **Apache**, входящий в состав пакета **httpd**. Для установки сервера Apache необходимо установить в системе пакет **httpd** или всю группу **web-server**, используя команду **yum groupinstall**. Следует иметь в виду, что в группу пакетов **Web-server** входит порядка 20 пакетов, включая **httpd** и **squid**.

Основным конфигурационным файлом веб-сервера Apache является файл **httpd.conf**, расположенный в каталоге **/etc/httpd/conf**. Кроме того, в этом каталоге имеется подкаталог **conf.d**, который содержит дополнительные конфигурационные файлы (например, конфигурационный файл **ssl.conf**, используемый для настройки защищенного SSL доступа к серверу Apache), которые включаются в основной конфигурационный файл.

Конфигурационные параметры, содержащиеся в файле **httpd.conf**, называются *директивами*. Каждая директива в файле **httpd.conf** записана в «контейнере». В начале «контейнера» указываются название директивы в угловых скобках **< >**, например, **<Directory "/var/www/icons">**. В конце «контейнера» указывается также название директив в угловых скобках, но в начале названия должен присутствовать символ **</>**, например, **</Directory>**.

Конфигурационный файл **httpd.conf** можно разделить на три основные секции, содержащие определенные группы директив.

К **глобальным директивам** относятся такие настройки, как корневой каталог сервера (директива **ServerRoot**), TCP-порт, через который HTTP-сервер должен принимать запросы (директива **Listen**), и параметры загрузки динамических мо-

дулей (директивы **LoadModule**).

В данной секции **основные директивы** настраиваются системные параметры сервера Apache. Здесь представлены такие конфигурационные параметры, как имя пользователя и группы, от имени которых будет запускаться сервер, важнейшая директива **DocumentRoot**, которая определяет корневой каталог обслуживаемых документов, и др. В этом разделе также задается обработка специальных URL-адресов, наподобие тех, что включают синтаксис **~имя_пользователя** для получения доступа к домашнему каталогу пользователя. Глобальные параметры безопасности тоже устанавливаются в данной секции конфигурационного файла. Некоторые директивы позволяют управлять доступом на уровне отдельных файлов (директива **File**) или на уровне отдельных каталогов (директива **Directory**). Именно с их помощью можно предотвратить доступ к важным файлам сервера Apache.

В следующем листинге приведен пример первых двух секций конфигурационного файла `http`

```
#Section 1. Global Environment
ServerRoot /etc/httpd
Listen 80
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
User apache
Group apache
#Section 2. 'Main' server configuration
ServerAdmin webmaster@example.com
ServerName example.com
DocumentRoot /var/www/html
DirectoryIndex index.html index.php index.txt
ErrorDocument 404 /errors/404.html
Options Indexes MultiViews
```

Основные директивы первых двух секций, на которые следует обратить внимание, **ServerRoot**, **Listen**, **ServerName** и **DocumentRoot**. В приведенном примере видно, что конфигурационные файлы веб-сервера Apache находятся в каталоге `/etc/httpd`; веб-сервер будет доступен по 80 порту; имя сервера, которое будет отображаться в ответах на клиентские запросы, будет **example.com**, а корневым каталогом, в котором содержатся данные, является `/var/www/html`.

Директивы виртуальных хостов

В данной секции конфигурационного файла настраиваются *виртуальные хосты*. Виртуальные хосты используются в тех случаях, когда на сервере Apache необходимо содержать несколько сайтов на одном IP-адресе. Существует два типа виртуальных хостов: основанные на имени и основанные на IP-адресе. В первом случае подразумевается, что на одном IP-адресе функционируют несколько различных сайтов; во втором случае каждый сайт функционирует на определенном IP-адресе.

Директивы, относящиеся ко всем трем секциям рассматриваемого конфигурационного файла, описаны в табл. 2.4.

Таблица 2.4. Директивы конфигурационного файла **httpd.conf**.

Директива	Описание
Первая секция	
ServerRoot	Определяет каталог, в котором хранятся конфигурационные файлы, сообщения об ошибках и журнальные файлы. По умолчанию данной директиве соответствует значение /etc/httpd .
Listen	Указывает порт, на котором веб-сервер ожидает входящих соединений. Возможно указание на определенный интерфейс, например 172.16.0.2:80 .
Timeout	Определяет количество времени в секундах, в течение которого сервер ожидает завершения событий, связанных с получением GET-запроса или получением tcp-пакетов при запросах PUT и POST. По умолчанию данное значение составляет 120 с.
KeepAlive	Если значение данной директивы соответствует « on », то каждый запрос обрабатывается только одним соединением.
LoadModule	Определяет использование дополнительных модулей Apache, например, mod_alias (предназначен для перенаправления запросов на другой адрес). Общая запись определения модуля имеет следующий вид: LoadModule <имф_модуля> modules/<название_файла_модуля>.so При использовании модуля alias данная директива будет иметь следующий вид: LoadModule alias_module modules/mod_alias.so
User и Group	Задаёт идентификаторы пользователя и группы, которые используются для запуска демона httpd. По умолчанию значению данной директиве соответствует пользователь и группа apache .
Вторая секция	
ServerAdmin	Содержит контактные данные администратора – URL и почтовый адрес.
ServerName	Имя хоста и порт, который используется сервером для собственной идентификации перед клиентами.
DocumentRoot	Определяет корневой каталог, в котором хранится все веб-содержимое сайтов, к которым предоставляет доступ данный веб-сервер.
DirectoryIndex	Содержит индексные файлы, которые могут использоваться для формирования корректных адресов, в случае использования запросов вида http://www.example.com/ . Обычно данная директива включает такие значения как index.html , index.php , index.txt . С учетом данных значений окончательный адрес будет иметь вид http://www.example.com/index.lhtm .

Директива	Описание
Directory	Определяются независимые каталоги, на основе которых можно управлять доступом к веб-содержимому.
Третья секция	
VirtualHost	Описывает виртуальные хосты, для которых возможно настроить разные политики доступа к веб-содержимому.
NameVirtualHost	Указывает IP-адрес, который будет использоваться для нескольких виртуальных хостов.
ServerAdmin	Присваивает почтовый ящик администратора указанного виртуального хоста.
DocumentRoot	Определяет корневой каталог веб-содержимого, используемый для виртуального хоста.
ServerName	Содержит название или URL виртуального хоста.
ErrorLog	Содержит файл с сообщениями об ошибках; путь указывается относительно каталога DocumentRoot.
CustomLog	Содержит файл с общими ошибками; путь указывается относительно каталога DocumentRoot.
VirtualHost	Описывает виртуальные хосты, для которых возможно настроить разные политики доступа к веб-содержимому каталогов.

В нижеследующем листинге приведен пример использования директивы **Directory** для указания различных *опций* каталогов, к которым осуществляется веб-доступ. Директива **Options** используется для указания опций, которые доступны для определенного каталога.

```
# Опции используемые для всех каталогов
<Directory />
Options FollowSymLinks
</Directory>
# Опции используемые для каталога DocumentRoot
<Directory "/var/www/html">
Options Indexes MultiViews
</Directory>
# Опции используемые для каталога /test
```

Как видно из листинга, имеются опции общие для всех подкаталогов, находящихся в корневом каталоге, которые определены в директиве **<Directory "/var/www/html">...</Directory>**. Если необходимо для определенного каталога настроить персональные опции, отличные от опций каталога **DocumentRoot**, то надо добавить еще одну директиву **Directory** и указать в ней необходимые опции. В табл. 2.5 указаны основные опции, используемые в директивах **Directory**, **File** или **VirtualHost**.

Таблица 2.5. Основные опции управления каталогами Apache

Опция	Описание
None	Не использовать никаких опций

Опция	Описание
All	Использовать все опции Apache, кроме MultiViews
ExecCGI	Разрешено выполнение CGI-скриптов
FollowSymLinks	Следовать символическим ссылкам внутри каталога
Includes	Разрешено использовать директиву include, используя модуль mod_include
Indexes	Использовать указанные индексы для каталогов
MultiViews	Позволяет клиенту отображать содержимое каталога, исходя из веб-браузера клиента, языка, предпочтительной кодировки и т.д.

Часто возникает необходимость иметь на одном веб-сервере Apache несколько независимых сайтов, используемых разными организациями. Для этого существует поддержка *виртуальных хостов* (virtual hosts). Виртуальные хосты делятся на два типа: именованные (**name-based**) и адресные (**IP-based**). Адресные виртуальные хосты используют IP-адрес сервера, при запросе соединения клиентом и на его основе определяют необходимый виртуальный хост, который следует «отдать» клиенту. Таким образом, у каждого сайта должен быть независимый IP-адрес. При использовании именованных виртуальных хостов с помощью директивы **NameVirtualHost** выделяется один IP-адрес для всех виртуальных хостов,. Затем для каждого виртуального хоста директивой **VirtualHost** определяется каталог на веб-сервере. В данной директиве обязательно должна присутствовать директива **ServerName**, в которой указывается DNS-имя ресурса, «отдаваемого» клиенту.

В следующем листинге приведен пример использования виртуального хостинга на основе адресных виртуальных хостов. В директиве **VirtualHost** определяются персональные настройки доступа к каталогу yum:

```
<VirtualHost 192.168.137.3:80>
    ServerAdmin root@localhost
    ServerName main.linux.lab
    Options Indexes FollowSymLinks
    DocumentRoot "/var/www/html/yum/"
    ErrorLog logs/main.linux.lab-error_log
    CustomLog logs/main.linux.lab-access_log common
</Virtualhost>
```

Из листинга видно, что в случае использования виртуальных хостов возможен более гибкий контроль доступа к содержимому каталога, к которому осуществляется веб-доступ. В частности, существует возможность выбора интерфейса и порта, на который необходимо принимать входящие HTTP-соединений, и указывать персональные журнальные файлы, содержащие информацию об ошибках.

В следующем листинге приведен пример использования виртуального хостинга на основе именованных виртуальных хостов. В директиве **NameVirtualHost** определяется IP-адрес, используемый для всех виртуальных хостов. В двух директивах **VirtualHost** указываются каталоги **site1** и **site2**, для которых необходимо настроить индивидуальный хостинг:

```
NameVirtualHost 192.168.0.1

<VirtualHost www.example.com>
ServerName www.example.com
ServerAdmin webmaster@example.com
DocumentRoot /var/www/html/site1
ErrorLog logs/error_log
TransferLog logs/access_log
</VirtualHost>

<VirtualHost company.com>
ServerName company.com
ServerAdmin webmaster@company.com
DocumentRoot /var/www/html/site2
ErrorLog logs/error_log
TransferLog logs/access_log
</VirtualHost>
```

После того как в конфигурационные файлы демона **httpd** внесены все необходимые изменения, необходимо проверить их синтаксис, выполнив команду **httpd -t**. Если в выводе данной команды отобразится сообщение “**Syntax OK**”, можно приступать к запуску демона **httpd**. Для этого можно использовать команду **service httpd start** или же использовать команду **apachectl**, входящую в состав пакета **httpd**.

2.5. Администрирование прокси-сервера Squid

При большой нагрузке на веб-сервер целесообразно использовать механизмы кэширования данных, при которых одни и те же данные будут доставляться клиенту не напрямую с веб-сервера, а из некоторого кэшированного хранилища данных. Одним из решений подобной проблемы является использование кэширующего *прокси-сервера Squid*.

Прокси-сервер не просто кэширует данные, получаемые в результате выполнения клиентских запросов, но и позволяет формировать иерархию прокси-серверов, которые могут обмениваться информацией о кэшированных данных между собой через прокол **ICP**, тем самым максимально оптимизируя процесс выборки кэшированных данных.

Помимо кэширования информации, прокси-сервер Squid может использоваться как ретранслятор (прозрачный или фильтрующий) запросов корпоративных пользователей в сеть Интернет. Это может быть выгодно с точки зрения учета интернет-трафика, контроля передаваемых данных и информационной безопасности внутреннего периметра организации.

В ОС Linux прокси-сервер Squid входит в состав группы пакетов **Web Server**. Его также можно установить отдельно из пакета **squid**.

После установки пакета **squid** в системе будут присутствовать следующие конфигурационные и бинарные файлы, которые используются прокси-сервером Squid:

- `/etc/init.d/squid` – init-скрипт запуска прокси-сервера Squid;
- `/etc/squid` – каталог, в котором содержатся все конфигурационные файлы прокси-сервера Squid;
- `/etc/sysconfig/squid` – файл, в котором содержатся опции запуска прокси-сервера Squid при помощи init-скрипта;
- `/usr/share/doc/squid-<версия>` – каталог с документацией в формате HTML;
- `/usr/lib/squid/` – каталог, содержащий специальные программы (**helpers**) используемые прокси-сервером Squid для аутентификации пользователей;
- `/usr/sbin/squid` – демон прокси-сервера Squid;
- `/usr/share/squid` – каталог, содержащий шаблоны сообщений об ошибках;
- `/var/log/squid` – каталог, в котором выполняется журналирование системных событий прокси-сервера Squid;
- `/var/spool/squid` – каталог, используемый для хранения кэшированных данных.

Общая последовательность действий для развертывания прокси-сервера Squid следующая:

1. Установка пакета `squid` и всех его зависимостей.
2. Настройка конфигурационного файла `squid.conf`.
3. Создание базы кэшированных данных.
4. Запуск демона `squid` и настройка его автозапуска.

Конфигурирование прокси-сервера Squid сводится в основном к настройке его конфигурационного файла `/etc/squid/squid.conf`, который вместе с комментариями содержит более 4000 строк. В каждой строке основного текста указывается определенная директива, имеющая несколько параметров. Остановимся на некоторых из директив.

Для того чтобы указать порт, который будет обрабатывать клиентские запросы, используется директива `http_port <номер_порта>`, в которой необходимо указать номер порта (он должен быть более 1024, поскольку демон `squid` запускается от не привилегированного пользователя).

Директива `hierarchy_stoplist` определяет условия, при которых запросы будут направляться напрямую веб-серверу, минуя кэш. Типовая директива `hierarchy_stoplist` имеет вид:

```
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
```

Данная директива помогает прокси-серверу Squid определить время хранения данных в кэше. Если время хранения данных превышено, то запрос будет сделан напрямую к веб-серверу для получения объекта. Данные, которые удовлетворяют заданным в данной директиве параметрам, отдаются клиенту из кэша. В следу-

ющем примере указывается время актуальности данных протокола FTP, которое составляет 1440 минут (один день).

```
refresh_pattern ^ftp:          1440    20%    10080
```

Если изначально исходным веб-сервером для данных FTP не было установлено максимальное время актуальности в заголовках **Expires** или **Cache-Control: max-age** и файл данных был помещен в кэш 10 часов назад, то коэффициент 20% означает, что данные будут считаться актуальными в течение следующих 2 часов. Максимальное время «актуальности» данных FTP в кэше прокси-сервера Squid в данном примере составляет 10080 минут (7 дней). После истечения максимального времени актуальности последующие запросы клиентов к этим данным будут направлены на веб-сервер.

Основной механизм фильтрации данных, проходящих через прокси-сервер, заключается в использовании правил, оперирующих списками доступа, заданными в директивах **acl**. В конфигурационном файле **squid.conf** изначально определены следующие списки доступа:

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
```

В первой строке указан список **all**, в состав которого входят все IP-адреса всех подсетей, во второй строке указан список доступа **manager**, которому удовлетворяет протокол **cache_object**. Далее идет определение списка доступа **localhost**, которому соответствует IP-адрес 127.0.0.1 с маской подсети 255.255.255.0. В последней строке представлен список доступа **to_localhost**, которому удовлетворяют все запросы с заголовком получателя, равным 127.0.0.0/8.

Следующие списки доступа определяют порты, через которые поступает трафик, который необходимо кэшировать:

```
acl SSL_ports port 443 563
acl Safe_ports port 80
acl Safe_ports port 21
acl Safe_ports port 443
acl Safe_ports port 70
acl Safe_ports port 210
acl Safe_ports port 1025-65535
```

После того, как в конфигурационном файле **squid.conf** были определены все необходимые списки доступа, можно определить правила, которые будут использоваться для фильтрации проходящего веб-трафика. Одним из таких правил является доступ по протоколу **http**, которое указывается в директиве **http_access**. В следующем примере первые две директивы разрешают доступ для списков **localhost** и **manager**. В последней директиве запрещается доступ по всем не указанным портам (**Safe_ports**).

```
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
```

Для того, чтобы запустить прокси-сервер Squid, в его конфигурационном файле должны обязательно быть определены следующие директивы:

- `visible_hostname <имя_хоста>` (задается имя прокси-сервера);
- `acl <список> <образец>` (определяется список доступа, соответствующий указанному образцу);
- `http_access allow <список_доступа>` (для указанного списка доступа разрешается доступ к данным по протоколу HTTP).

После того как конфигурационный файл `squid.conf` будет сформирован, его необходимо проверить на наличие ошибок при помощи команды **squid -k parse**. Если вывод данной команды окажется пустым, ошибки отсутствуют. В случае наличия ошибок будет указана соответствующая строка и выведено описание ошибки.

Перед запуском демона squid необходимо создать базу данных кеша, с помощью команды **squid -z**. После выполнения данной команды в каталоге `/var/cache/squid` будут созданы все необходимые файлы базы данных кеша.

Наконец, когда база данных кеша будет создана, можно запустить демон **squid** при помощи команды **service squid start** и добавить его в автозапуск, используя команду **chkconfig**.

Если в ОС Linux активизирован механизм SELinux, то для того, чтобы прокси-сервер Squid мог обмениваться данными по сети, необходимо установить значение параметра **squid_connect_any** равное 1:

```
setsebool -P squid_connect_any 1
```

2.6. Защищенное администрирование. Пакет OpenSSH

Интерпретатор **ssh (secure shell)** является заменой небезопасному интерпретатору **telnet**. Он использует защищенную аутентификацию для подтверждения личности пользователя и шифрует любые соединения между двумя хостами. Интерпретатор **ssh** входит в состав пакета **openssh-clients**, который также содержит команду безопасного копирования файлов **scp** и команду защищенного копирования файлов по протоколу FTP – **sftp**.

В ОС Linux версия защищенного командного интерпретатора **ssh** называется **OpenSSH**. Принцип работы OpenSSH основывается на архитектуре «клиент-сервер». Система, к которой необходимо подключиться, выступает в качестве сервера. В данной системе функционирует демон **sshd**, который обслуживает удаленные подключения. Система, которая подключается к демону **sshd**, выступает в качестве клиента. Помимо базовых возможностей, таких как работа в командной строке, интерпретатор **ssh** позволяет организовывать безопасную трансляцию сеансов X Window с удаленного хоста на хост администратора, позволяя тем самым работать в графической среде удаленной системы при помощи локального терминала. Еще одной замечательной возможностью, которой обладает данный интерпретатор, яв-

ляется перенаправление портов (**port forwarding**), при помощи которого возможно перенаправлять соединения к серверу, на котором настроено перенаправление, на удаленный сервер, который используется для конечной обработки запроса.

Для организации сервера OpenSSH в ОС Linux необходимо установить пакет **openssh-server**. В процессе его инсталляции будет создана пара ключей – закрытый и открытый. Открытый ключ используется клиентами для подтверждения подлинности данного сервера. Все сообщения между клиентом и сервером шифруются открытым ключом сервера. Закрытый ключ используется для расшифрования сообщений на сервере. После переустановки ОС или смены IP-адреса на сервере OpenSSH клиенту будет выведено уведомление о несоответствии открытого ключа сервера тому ключу, который был передан клиенту изначально. Информация об открытых ключах содержится в файле `~/.ssh/known_hosts`.

В состав клиентской части OpenSSH входят следующие утилиты:

- **ssh-agent** – предназначена для хранения закрытых ключей, используемых для аутентификации RSA. В начале каждого графического или консольного сеанса утилита запускается, и все остальные окна или консольные утилиты работают как клиенты по отношению к ней;
- **ssh-add** – используется для передачи идентификационных данных RSA утилите `ssh-agent`;
- **ssh-keygen** – используется для генерации закрытого и открытого ключей. Например, для создания ключей типа RSA необходимо выполнить команду **ssh-keygen -t rsa**;
- **scp** – используется для защищенного копирования данных, подобного команде `cp`;
- **sftp** – используется для защищенной передачи файлов;
- **ssh** – защищенный интерпретатор команд.

Настройка серверной части OpenSSH достаточно проста и заключается в редактировании конфигурационного файла демона `sshd` – `/etc/ssh/sshd_config`. В данном файле следует обратить внимание на следующие директивы.

- **SyslogFacility AUTHPRIV**, в которой директиве задается уровень детализации сообщений, направляемых в файл `/var/log/secure`. В процессе тестирования соединений SSH полезно использовать данный файл для просмотра системных событий;
- **PasswordAuthentication = yes|no** – в ней указывается, что сервер поддерживает аутентификацию в интерактивном режиме при помощи паролей;
- **X11Forwarding yes|no**, позволяющая перенаправлять сеансы X с данного хоста на хост клиента. Для подключения к сенсу удаленного хоста необходимо использовать команду **ssh -X <пользователь>@<удаленный_хост>**, в которой указывается имя пользователя удаленного хоста, а также имя или IP-адрес самого хоста;
- **PermitRootLogin yes|no**, которая определяет возможность подключения к

серверу, используя учетную запись пользователя **root**;

- **RSAAuthentication yes|no**, определяющая поддержку аутентификации с использованием алгоритма RSA;
- **PubkeyAuthentication yes|no**, в которой определяется поддержка сервером аутентификации на основе открытого ключа клиента. Данный механизм аутентификации более надежен, чем аутентификация по паролю. Она позволяет использовать ssh для написания всевозможных административных сценариев, где необходимо выполнять вход на удаленный хост без ввода пароля;
- **AuthorizedKeysFile.ssh/authorized_keys**, определяющая файл, в котором содержатся открытые ключи клиентов. Определяемый файл и его родительский каталог должны иметь код доступа **700** для того, чтобы было возможным использовать аутентификацию по открытому ключу.

После того как серверная часть настроена, необходимо запустить демон **sshd**, используя команду **service**.

Настройки клиентской части OpenSSH содержатся в конфигурационном файле **/etc/ssh/ssh_config**, который содержит директиву **Host**, определяющую применимость всех директив данного файла к указанному хосту, а также некоторые директивы, присутствующие в файле **sshd_config**.

К основным операциям относятся следующие.

- **Подключение к удаленному хосту** с использованием следующей команды:

```
ssh <имя_пользователя>@<имя_хоста> ,
```

где в качестве аргумента **<имя_пользователя>** необходимо ввести соответствующее регистрационное имя удаленного пользователя, а в аргументе **<имя_хоста>** указать DNS имя или IP-адрес удаленного хоста.

- **Выполнение команды на удаленном хосте**. Для ее запуска используется следующая команда:

```
ssh <имя_пользователя>@<имя_хоста> <команда> ,
```

где в качестве аргумента **<команда>** необходимо указать нужную команду. Следует отметить, что бинарный файл, указанной в данной команде, должен существовать на удаленном сервере.

- **Передача файлов на удаленный хост**. Данная команда имеет следующий синтаксис:

```
ssh <локальный_файл> <имя_пользователя>@<имя_хоста>:<удаленный_файл>
```

Вместо аргумента **<локальный_файл>** и **<удаленный_файл>** необходимо, соответственно, указать локальный и удаленный файл.

- **Перенаправление портов** позволяет шифровать данные, передаваемые по незащищенным портам. Например, клиент, находящийся в сети Интернет, может забирать свои почтовые сообщения по протоколу POP3 с почтового сервера, находящегося в локальной сети организации и при этом быть уве-

рен, что передаваемые им данные защищены.

Основной синтаксис команды, используемой для перенаправления локальных портов, следующий:

```
ssh -L <локальный_порт>:<удаленный_хост>:<удаленный_порт>  
<имя_пользователя>@<удаленный_хост>
```

Когда клиент выполняет подключение к серверу на локальный порт, соединение перенаправляется на удаленный порт удаленного сервера. Далее выполняется аутентификация клиента на основе имени пользователя и хоста, который содержит его учетную запись. При этом формируется зашифрованный туннель между клиентом и удаленным хостом.

2.7. Система обмена почтовыми сообщениями. Хранилище почтовых данных

Современная почтовая система рассматривается не просто как агент передачи почтовых сообщений (Mail Transfer Agent (MTA)), выполняющий простую доставку писем, а как комплекс коллективных средств взаимодействия пользователей, в который могут входить календари, адресные книги, средства приема телефонных звонков (VoIP) и мгновенных сообщений (IM). Для отправки и приема почтовых сообщений используются специальные почтовые клиенты (Mail User Agent (MUA)), которые подключаются к специальной программе, работающей совместно с MTA и предоставляющей доступ к почтовым ящикам по протоколам POP3, IMAP или IMAPS.

В ОС Linux имеется несколько почтовых агентов на выбор, в частности **Sendmail** и **Postfix**. Первый из них используется в системе по умолчанию. Что касается программы предоставления доступа к почтовым ящикам пользователей, то оптимальным вариантом в данном случае будет являться ПО **Dovecot**. В данном разделе мы сосредоточимся на изучении MTA Sendmail и Dovecot.

2.7.1. Конфигурирование Sendmail

Чтобы организовать в ОС Linux сервис передачи почтовых сообщений, в системе должны быть установлены пакеты **sendmail**, **sendmail-cf** и **sendmail-doc**. В первом пакете содержится собственно MTA Sendmail, предназначенный для обработки SMTP-соединений. Во втором пакете содержатся средства, позволяющие изменять конфигурационный файл MTA Sendmail. Третий пакет содержит исчерпывающую документацию по настройке MTA Sendmail.

В целях экономии времени на экзамене, данные пакеты можно установить за раз, выполнив установку в системе группы пакетов **Mail Server**. Лучше всего выполнять данную процедуру, используя менеджер пакетов **yum**. Помимо перечисленных пакетов при установке группы пакетов **Mail Server** в систему будут установлены пакеты **dovecot** (программа доступа к почтовым ящикам), **cyrus-sasl**

(программа, позволяющая использовать расширенный механизм аутентификации SASL) и **spamassassin** (программа, предназначенная для борьбы с нежелательными сообщениями).

Основную работу по обработке почтовых сообщений, передаваемых по протоколу SMTP, выполняет демон **sendmail**, конфигурационные файлы которого располагаются в каталоге **/etc/mail**. В данном каталоге находятся два основных конфигурационных файла: **sendmail.cf**, использующийся для обработки входящей почты, и **submit.cf**, использующийся для обработки исходящей почты. Кроме того каталог **/etc/mail** содержит следующие важные файлы:

- **sendmail.mc** – содержит макросы **sendmail**; используется для изменения конфигурации, записанной в файле **sendmail.cf**;
- **submit.mc** – также содержит макросы **sendmail**, но используется для изменения конфигурации, записанной в файле **submit.cf**;
- **access**, определяющий ограничения на отправку исходящей почты. По умолчанию разрешена отправка исходящей почты только с локального хоста. В данный файл необходимо добавлять имена хостов или IP-адреса подсетей с параметрами **REJECT**, **DISCARD** и **RELAY**. При указании параметра **REJECT** хосту-отправителю будет выдано сообщение об ошибке. Во втором случае сообщение выдаваться не будет, но доступ также будет закрыт. При указании параметра **RELAY** указанный хост сможет выполнять отправку исходящих сообщений;
- **domaintable** – в нем указываются почтовые домены, почту для которых следует отправлять на альтернативный почтовый домен. например:

```
example.com company.com
```

В данном случае почту адресованную пользователю **user1@example.com** возможно перенаправлять на почтовый ящик **user1@company.com**;

- **mailertable** – в данном файле возможно указать транспорт (mailer) и почтовый сервер, куда будут направляться почтовые сообщения предназначенные для определенного почтового домена. По умолчанию для маршрутизации почтовых сообщений MTA Sendmail использует MX-записи почтовых доменов на DNS-серверах. Чтобы использовать данный файл, в файле **sendmail.mc** должен быть включен макрос **FEATURE(`mailertable');**
- **virtusertable** – используется в основном для перенаправления почты от конкретного пользователя в указанный почтовый ящик;
- **Makefile** – используется для компиляции файла **sendmail.mc**. Конфигурационный файл **sendmail.cf** не рекомендуется редактировать напрямую по причине его специального формата. Для изменения конфигурации сначала необходимо отредактировать файл **sendmail.mc** а затем запустить команду **make sendmail.mc > sendmail.cf**;
- **statistic**, в котором собирается статистика об использовании MTA Sendmail в бинарном формате. Для его просмотра используется команда **mailstats**.

MTA Sendmail работает с конфигурационными файлами не напрямую, а через их модифицированную копию, имеющую специальный формат, например, **hash**. Поэтому если в файлы **access**, **domaintable**, **mailertable** или **virtusertable** были внесены изменения, то необходимо использовать команду **makemap**, генерирующую данные файлы. Следует также отметить, что при запуске, перезапуске или перечитывании демона **sendmail** данная команда выполняется автоматически.

В дополнение к файлу **virtusertable** в ОС Linux для перенаправления почтовых сообщений от одного пользователя к другому используется файл **/etc/aliases**. По умолчанию в данном файле содержатся правила перенаправления почты системных пользователей пользователю **root**. Данный файл целесообразно использовать для перенаправления всей почты, адресованной пользователю **root**, на почтовый ящик администратора. После модификации данного файла демон **sendmail** должен быть перезапущен, чтобы перечитать данный файл и сформировать его оптимизированный вариант с помощью команды **newaliases**.

По умолчанию MTA Sendmail настроен на прием входящих соединений только с локального IP-адреса 127.0.0.1. Для того, чтобы демон **sendmail** смог принимать внешние соединения в файл **/etc/mail/sendmail.mc**, необходимо найти следующий макрос и изменить параметр **Addr** либо закоментировать данную строку, добавив в начало строки слово **dnl** и символ пробела:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

Директива **FEATURE** используется для добавления в MTA Sendmail дополнительной функциональности, например, макрос **FEATURE(`accept_unresolvable_domains')dnl** используется для приема почтовых сообщений от доменов, доменные имена которых невозможно разрешить в DNS.

После того как конфигурационный файл **sendmail.mc** будет отредактирован, необходимо добавить в файл **/etc/mail/access** список удаленных хостов, которым будет разрешено отправлять почту через данный почтовый сервер. Пример такой записи имеет следующий вид:

```
192.168.0          RELAY
```

В данном примере для всех хостов в подсети 192.168.0 будет разрешено отправлять письма через данный сервер.

Чтобы сделанные настройки вступили в силу, необходимо сгенерировать файл **sendmail.cf**, а также преобразовать все измененные файлы в каталоге **/etc/mail**, такие как **access** и **mailertable**, в специальный формат, понятный для демона **sendmail**. Для экономии времени можно запустить единственную команду **make -C /etc/mail**, которая выполнит все необходимые действия и перезапустит демон **sendmail**.

MTA **Sendmail** может доставлять почту только в локальные ящики пользователей, находящихся в каталоге **/var/spool/mail** или пересылать ее другим хостам. В данном случае почтовые ящики пользователей имеют формат **MAILBOX** и представляют собой обычные текстовые файлы, в которых каждое письмо отделено пустой строкой.

Начиная с версии Sendmail 8.12¹ сообщения поступают в почтовую систему, используя каталог `/var/spool/clientmqueue`, руководствуясь настройками файла `submit.cf`.

Программа **sendmail** позволяет создавать сразу несколько почтовых очередей и объединять очереди в группы. Если, к примеру, каталог **mqueue** содержит подкаталоги `q1`, `q2` и `q3` и каталог очереди задается как `/var/spool/mqueue/q*`, то будут использоваться все три очереди. Это способствует повышению производительности при высокой загрузке системы.

Группы очередей присутствуют во всех версиях Sendmail начиная с 8.12. Они позволяют четко управлять доставкой отдельных типов сообщений. Любой параметр, связанный с очередью, включая приоритет выполнения (назначается с помощью команды **nice**), может быть задан и для группы очередей. Почтовые сообщения направляются в ту или иную группу в зависимости от адреса самого первого получателя сообщения. Стандартная группа называется **mqueue**; она определяется автоматически и доступна по умолчанию, не требуя никакого конфигурирования.

2.7.2. Конфигурирование Dovecot

После того, как агент передачи почтовых сообщений Sendmail будет настроен и запущен, следует приступить к настройке ПО **Dovecot**, которое обеспечивает доступ пользователей к своим почтовым ящикам по протоколу IMAP, POP3 или их защищенным аналогам - IMAPS и POP3S.

Прежде чем настраивать данное ПО, необходимо установить пакет **dovecot**, если таковой отсутствует. Основным конфигурационным файлом данного ПО является файл `/etc/dovecot.conf`. Для поддержки желаемых протоколов доступа необходимо раскомментировать следующую директиву:

```
protocols = imap imaps pop3 pop3s
```

Если на сервере установлено несколько сетевых интерфейсов, то в директиве `listen` или `ssl listen` необходимо указать IP-адрес и порт, которые будут использоваться для подключений клиентов:

```
listen = 192.168.0.1:10110
ssl listen = 192.168.0.2:10943
```

Если используются защищенные протоколы IMAPS или POP3S, необходимо также раскомментировать следующие директивы:

```
ssl_cert_file = /etc/pki/dovecot/certs/dovecot.pem
ssl_key_file = /etc/pki/dovecot/private/dovecot.pem
```

Расположение почтовых ящиков пользователей указывается в директиве **mail_location**. Данную директиву обязательно необходимо заполнить, иначе могут возникнуть проблемы с доставкой почты пользователям, которые не имеют отдельных почтовых каталогов в каталоге `/var/spool/mail`. Типовой пример ука-

¹ Для того чтобы узнать текущую версию MTA Sendmail необходимо запустить команду **sendmail** с ключом **-d0.1**.

зания места расположения пользовательских ящиков имеет следующий вид:

```
mail_location = maildir:~/Maildir
```

В данном случае используется формат почтовых ящиков **MAILDIR**, который, в отличие от формата **MAILBOX**, располагает сообщения пользователей в отдельных каталогах. С точки зрения производительности, данный формат рекомендуется использовать по умолчанию.

После того как демон **dovecot** будет настроен и запущен, необходимо убедиться, что он ожидает входящих соединений на указанном порте. Для этого можно воспользоваться штатной утилитой **telnet**.

2.8. Администрирование общих сетевых сервисов (xinetd, FTP, NTP)

В данном разделе описываются дополнительные сетевые сервисы, которые используются как самостоятельно, так и при конфигурировании основных сетевых сервисов.

2.8.1. Сервис xinetd

Не все серверы имеют собственные процессы-демоны, которые обрабатывают сетевые подключения клиентов. Для решения данной проблемы в ОС Linux присутствует демон **xinetd**.

Демон **xinetd** – это демон, который, по сути, управляет другими демонами. Он запускает своих демонов-клиентов только тогда, когда к ним происходит обращение, и позволяет им корректно завершать свою работу сразу после того, как те выполнят все свои задачи. Демон **xinetd** отвечает за функционирование множества распространенных сетевых сервисов, поэтому он играет важную роль в обеспечении безопасности системы. Для его установки необходимо установить пакет **xinetd**.

Основным конфигурационным файлом демона **xinetd** по умолчанию является файл **/etc/xinetd.conf**. Помимо данного файла в ОС Linux имеется каталог **/etc/xinetd.d**, куда некоторые пакеты помещают свои конфигурационные файлы, и не засоряют основной конфигурационный файл. Настройки, хранящиеся в данном каталоге, имеют более высокий приоритет, чем те же самые настройки, присутствующие в файле **xinetd.conf**.

Ниже приведен пример конфигурационного файла **xinetd.conf**, используемого по умолчанию:

```
defaults
{
    log_type           = SYSLOG daemon info
    log_on_failure     = HOST
    log_on_success     = PID HOST DURATION EXIT
    cps                = 50 10
    instances          = 50
    per_source         = 10
```

```

        v6only          = no
        groups          = yes
        umask           = 002
    }
    includedir /etc/xinetd.d

```

В данном примере настройки по умолчанию задаются в директиве **defaults**. В директиве **log_type** указывается метод журналирования событий. Как видно из данного примера, в директиве **log_type** для журналирования событий используется сервис **syslog**. События посылаются демону **syslogd** с помощью средства **daemon** и с уровнем детализации **info**. Директива **cps** определяет максимальное число соединений в секунду. Если с сервером устанавливается более некоторого предельного числа соединений в секунду (в данном примере – 50), то все последующие соединения будут отклоняться. Кроме того, данная директива определяет, что демон **xinetd** будет ожидать в течение 10 секунд, прежде чем опять принять входящие соединения от клиента. В директиве **instances** определяется максимальное количество процессов демона **xinetd**, которое может обслуживать клиентские запросы. В директиве **per_source** накладывается ограничение на количество соединений с одного IP-адреса. Директивы **groups** и **umask** позволяют выполнять запуск демона **xinetd** от непривилегированного пользователя. В конце файла **xinetd.conf** указана директива **includedir**, включающая в состав конфигурационного файла дополнительные файлы из каталога **xinetd.conf**.

В следующем листинге приведен пример конфигурационного файла **xinetd** для сервиса **rsync**:

```

# default: off
service rsync
{
    disable = yes
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = -daemon
    log_on_failure += USERID
}

```

Основными директивами в данном примере являются **disable**, **user** и **server**. В директиве **disabled** определяется, будет ли сервис использовать для обслуживания соединений демон **xinetd**. В директиве **user** указывается регистрационное имя пользователя, UID которого будет присвоен порожденному процессу. В директиве **server** указана команда, которая будет выполняться при каждом обращении к демону **xinetd**. Все остальные директивы, используемые в данном файле, можно посмотреть на странице руководства **man** для файла **xinetd.conf**.

2.8.2. Сервис FTP

В ОС Linux сервис FTP организуется на основе ПО **VSFTPD**. Для его использования в системе необходимо установить пакет **vsftpd**. Основным configura-

ционным файлом, который использует демон **vsftpd**, является файл **/etc/vsftpd.conf**. Используя данный файл, можно настроить такие параметры, как приветственное входное приглашения (**ftp banner**), разрешения на выгрузку и загрузку файлов, а также порт, на котором демон **vsftpd** ожидает входящих соединений.

Полный список всех директив данного конфигурационного файла содержится на страницах руководства man. В табл. 2.6 представлены основные директивы, используемые в процессе администрирования демона **vsftpd**. В круглых скобках указано значение, используемое по умолчанию.

Таблица 2.6. Основные директивы конфигурационного файла **vsftpd.conf**.

Директива	Описание
<code>listen_port</code> (21)	Порт, используемый для обработки входящих FTP-запросов
<code>ftp_banner</code> (отсутствует)	Приветствие, которое выводится на терминал после аутентификации на FTP-сервере
<code>local_enable</code> (NO)	Определяет, могут ли локальные пользователи аутентифицироваться на сервере FTP
<code>hide_ids</code> (NO)	Следовать символическим ссылкам внутри каталога
<code>max_clients</code> (0)	Разрешено использовать директиву <code>include</code> , используя модуль <code>mod_include</code>
<code>anonymous_enable</code> (YES)	Разрешить доступ для анонимных пользователей

Демон **vsftpd** имеет достаточно простой механизм контроля доступа пользователей. Для того, чтобы запретить определенному пользователю подключаться к серверу по протоколу FTP, его достаточно занести в файл **/etc/vsftpd/user_list**. По умолчанию системные пользователи, такие как `root`, занесены в данный файл. Если в файле **vsftpd.conf** определена директива **userlist_enable** со значением **YES**, то файл **/etc/vsftpd/user_list** просматривается для определения доступа пользователей к FTP-серверу. Если в файле **vsftpd.conf** определена директива **userlist_deny** со значением **YES**, то пользователи, перечисленные в файле **/etc/vsftpd/user_list**, не имеют доступа к FTP серверу. Если в файле **vsftpd.conf** определена директива **userlist_deny** со значением **NO**, то к ресурсам FTP-сервера будут допущены только пользователи, присутствующие в файле **/etc/vsftpd/user_list**.

Для подключения к серверу FTP используются команды **ftp** и **lftp**. Последняя имеет расширенные возможности: автозавершение имен файлов, отображение процесса загрузки файлов, загрузка непольностью загруженных файлов, автоматическое использование анонимного пользователя при подключении. Для того, чтобы подключиться к FTP-серверу, используется следующая команда:

```
lftp -u <имя_пользователя> <имя_сервера>
```

в качестве параметра **<имя_пользователя>** аргумента **-u** необходимо указать имя пользователя, которое будет использоваться при подключении к серверу FTP, а в качестве аргумента **<имя_сервера>** указать имя FTP-сервера. Для отображения всех поддерживаемых команд клиентом **lftp** необходимо ввести команду **help**.

2.8.3. Сервис NTP

В ОС Linux демон **ntpd** реализует протокол **NTP** (Network Time Protocol – протокол сетевого времени), который позволяет хостам синхронизировать свои часы между собой с точностью до миллисекунд.

Демон **ntpd** реализует как клиентскую, так и серверную часть протокола NTP. При запуске он считывает конфигурационный файл **/etc/ntp.conf**, в котором указываются права доступа, клиентские сети, службы времени, параметры общей конфигурации и параметры аутентификации.

Для настройки клиентской части протокола NTP, помимо редактирования файла **/etc/ntp.conf**, можно использовать графическую утилиту **Date/Time**, запустив команду **system-config-date**.

Конфигурационный файл **/etc/ntp.conf** должен содержать как минимум следующие строки:

```
restrict <имя_сервера> mask 255.255.255.255 nomodify notrap noquery  
server <имя_сервера>
```

В первой строке описывается директива ограничений **restrict**, которая налагает ограничения **nomodify**, **notrap** и **noquery** для сервера NTP, указанного в аргументе **<имя_сервера>**. Параметры **nomodify**, **notrap** и **noquery** означают, что настройки NTP сервера не могут быть изменены, контроль сообщений (message trap) невозможен и все запросы синхронизации запрещены. После настройки данного файла необходимо синхронизировать текущее время с сервером NTP, используя команду **ntpd -q <имя_сервера>**, и перезапустить сервис **ntpd**.

Если текущее время отличается от времени, полученного демоном **ntpd** после синхронизации более чем на 1000 секунд, демон **ntpd** завершит свою работу и не изменит текущее время. Для того чтобы принудительно синхронизировать время, необходимо использовать команду **ntpd -g -q <имя_сервера>**. Для того чтобы NTP-клиент стал работать в качестве сервера и клиенты могли определить его параметры без явного указания, необходимо настроить широковещательную рассылку времени, добавив в файл **/etc/ntp.conf** строку **broadcast 224.0.1.1 ttl 4**:

```
restrict default nomodify notrap noquery  
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap  
broadcast 224.0.1.1 ttl 4
```

В первой строке листинга определено запрещение по умолчанию для всех подсетей. Для того, чтобы разрешить конкретной подсети обращаться к данному NTP-серверу, используется вторая строка, в которой для подсети **192.168.0.0** разрешена синхронизация времени (отсутствует параметр **noquery**). В конце файла указан широковещательный адрес для автоматической настройки клиентов.

После изменения конфигурационного файла **/etc/ntp.conf** необходимо перезапустить демон **ntpd**, используя команду **service**.

2.9. Основы маршрутизации в сетях TCP/IP

В настоящее время к информационным системам предъявляются высокие требования по скорости передачи информации, надежности и масштабируемости. Связующим звеном в любой информационной системе является сетевая инфраструктура. От того как построена сетевая инфраструктура будет зависеть и производительность сетевых приложений. В процессе роста информационной системы непременно увеличивается количество хостов в сети и прочего сетевого оборудования. В связи с этим становится актуальным вопрос эффективной маршрутизации данных. Для выполнения задач маршрутизации используются специальное сетевое оборудование – маршрутизаторы. *Маршрутизатор* – это устройство, управляющее трафиком. Данные, передаваемые по IP-сетям, да и по любым другим сетям, могут распространяться самыми разными путями.

Все данные формируются в *пакеты* – отдельные блоки, посылаемые в сеть. В общем случае задача маршрутизатора состоит в выборе наилучшего из текущих маршрутов, следуя которому заданный пакет попадет в пункт своего назначения.

Способ принятия такого решения зависит от используемого протокола маршрутизации. У каждого протокола свой алгоритм слежения затем, какие маршруты доступны и какие из них наиболее эффективны. В частности, демон, работающий по протоколу одноадресной маршрутизации, посылает информацию непосредственно тем маршрутизаторам, на общение с которыми он сконфигурирован. Протоколы этого класса эксплуатируются в тех сетях, где очень важна пропускная способность.

Часто маршрутизаторы представляют собой программно-аппаратные комплексы, имеющие специальную операционную систему. Альтернативным вариантом решения задач маршрутизации является использование программного маршрутизатора на основе ОС Linux, который при соответствующей настройке может даже превзойти по производительности и функционалу традиционные программно-аппаратные маршрутизаторы. Для начала определим, какие типы маршрутизации существуют в сетях.

Простейшей формой маршрутизации является *статическая маршрутизация*. В этом случае создается обычная маршрутная таблица, которая остается неизменной до тех пор, пока не будет выдана соответствующая команда. Программы не пытаются проверять истинность маршрутов, приведенных в таблице, и не пытаются отслеживать изменения в топологии сети. Все это нужно делать вручную. Для настройки статической маршрутизации в ОС Linux используются утилиты **ip** или **route**. В самом простом случае, когда требуется объединить несколько подсетей, подойдет использование шлюза с включенным перенаправлением IP-пакетов.

Для включения перенаправления пакетов необходимо изменить параметр ядра, используя следующую команду:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Для сохранения сделанного изменения необходимо добавить запись **net.ipv4.**

`ip_forward = 1` в файл `/etc/sysctl.conf`. После добавления записи понадобится считывание файла при помощи команды `sysctl -p`. После того, как перенаправление пакетов будет активировано необходимо определить шлюзы по умолчанию для каждого сетевого интерфейса. Определение шлюза по умолчанию выполняется командой:

```
# route add default gw <IP-адрес-интерфейса> <имя_интерфейса>
```

или командой

```
# ip route add default via <IP-адрес-интерфейса> dev <имя_интерфейса>
```

Отличием двух команд является то, что команда `route` позволяет определить более одно маршрута по умолчанию.

В общем случае синтаксис команды `route` следующий:

```
route [-A <протокол>] add [-net|-host] <цель> [netmask <маска>] [gw <шлюз>] [dev <интерфейс>]
```

```
route [-A <семейство>] del [-net|-host] <цель> [dev <интерфейс>]
```

Здесь:

- **A <протокол>** – определяет тип используемого протокола (inet, inet6, ipx, x25 и прочие);
- **add/del** – добавляет или удаляет маршрут;
- **[-net | -host]** – используется для указание типа маршрута (сетевой маршрут или маршрут до хоста);
- **цель** – IP-адрес сети ли хоста, для которого создается маршрут;
- **netmask <маска>** – сетевая маска подсети или хоста;
- **gw <шлюз>** – IP-адрес интерфейса, через который следует выполнять маршрутизацию пакетов;
- **dev <интерфейс>** – сетевой интерфейс, на котором следует создать или удалить маршрут.

Для просмотра текущей таблицы маршрутизации используется команда `route` без параметров. В следующем листинге приведена типовая таблица маршрутизации:

```
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.1.0 * 255.255.255.0 U 0 0 0 eth1
172.16.0.0 * 255.255.255.0 U 0 0 0 eth0
169.254.0.0 * 255.255.0.0 U 0 0 0 eth1
default 192.168.1.254 0.0.0.0 UG 0 0 0 eth1
```

Данная таблица маршрутизации имеет следующие поля:

- `Destination` – хост или подсеть назначения;
- `Gateway` – IP-адрес шлюза или * если шлюз не указан;
- `Genmask` – Сетевая маска сети назначения: `255.255.255.255` – в случае хоста и `0.0.0.0` – в случае маршрута по умолчанию.
- `Flags` – флаги:

- U – маршрут активен;
- H – назначением маршрута является хост;
- G – использовать шлюз;
- R – восстановить маршрут для динамической маршрутизации;
- D – динамический маршрут или перенаправление;
- M – измененный маршрут.

В более крупных сетях, где количество узлов достаточно большое используется **динамическая маршрутизация**, позволяющая находить кратчайшие маршруты до места назначения пакетов.



Модуль 3. Организация информационной безопасности

После завершения изучения данного модуля вы научитесь:

- понимать общие принципы работы механизма SELinux;
- настраивать межсетевой экран на основе iptables;
- организовывать простейший VPN туннель;
- настраивать аудит системных событий.

3.1. Управление интеллектуальными списками доступа (SELinux)

Модель разграничения доступа к данным в ОС Linux имеет свои недостатки. Такая модель (концепция доступа к файлам по усмотрению учетной записи с соответствующими разрешениями) традиционно основывается на доверии. Предполагается, что пользователи с правами доступа не имеют злого умысла, что администраторы знают, какие разрешения должны быть у каждого файла в пакете программ.

Механизм **SELinux** решает эту проблему путем использования мандатного контроля доступа (Mandatory Access Controls – MAC). При мандатном контроле доступа пользователи не имеют полного права управлять доступом к объектам. Вместо этого администратор определяет общесистемные политики доступа. Хорошо реализованная политика MAC заключается в следовании принципу «наименьшего уровня привилегий», т.е. разрешения доступа только тогда, когда это действительно необходимо приложению. MAC может предотвращать нарушение защиты системы ПО с уязвимыми местами в коде выполнения (например, переполнениями буфера), ограничивая масштаб уязвимости несколькими конкретными, требующимися этому ПО ресурсами.

Разработка политик¹ SELinux является сложной темой. Например, для защиты нового демона в политике должны быть тщательно перечислены все файлы, каталоги и другие объекты, доступ к которым может быть необходим этому процессу.

¹ Большинство политик SELinux доступны в сети Интернет. Они легко устанавливаются и настраиваются при помощи графического редактора политик SEEdit, доступного по адресу <http://seedit.sourceforge.net/>.

3.1.1. Режим работы и политика механизма SELinux

В ОС Linux доступны следующие режимы работы механизма SELinux.

- **Enforcing.** Установка данного значения гарантирует применение политики SELinux и предотвращение несанкционированного доступа к файлам. При загрузке ОС с данным режимом работы существует возможность изменять его без перезагрузки системы.
- **Permissive.** При установке данного значения попытки нарушения политики допускаются, но при этом регистрируется в системе syslog.
- **Disabled.** При этом режиме механизм SELinux полностью отключается. При смене данного режима работы на любой другой необходимо заново пометить файловую систему в процессе перезагрузки системы.

После того как ОС Linux будет загружена впервые после инсталляции, утилита **Setup Agent** по умолчанию предложит использовать режим работы **Enforcing**.

Режим работы механизма SELinux задается в файле `/etc/selinux/config` в директиве **SELINUX**. Для смены режима работы механизма SELinux необходимо отредактировать данный файл, а затем использовать команду **setenforce <режим_работы>**. Для подтверждения сделанных изменений необходимо запустить команду **getenforce**, которая отображает текущий режим работы механизма SELinux. В ОС Linux имеется графическая утилита **SELinux Management Tool**, которая запускается при помощи команды **system-config-selinux**. После того как система перезагрузится, можно убедиться в смене режима, используя команду **sestatus**. Если выбран режим работы механизма SELinux `permissive` или `enforcing`, то необходимо определить **тип политики** SELinux. В ОС Linux доступны следующие основные политики SELinux:

- **Targeted.** Данный тип политики используется по умолчанию и направлен на защиту важных сетевых сервисов, таких как NFS, DNS, HTTP. Для использования данного типа политики требуется установка пакета **selinux-policy-targeted**.
- **Strict** – более жесткая, чем предыдущий тип. Пользователям разрешен доступ только к определенным каталогам, а процессам разрешен доступ только к необходимым каталогам. Для использования данной политики требуется установка пакета **selinux-policy-strict**.

Для того чтобы сменить политику, сначала необходимо установить соответствующий пакет, а затем в файле `/etc/selinux/config` указать требуемую политику в директиве **SELINUXTYPE**. После смены политики необходимо перезагрузить систему. В целях снижения риска некорректной работы механизма SELinux для начала рекомендуется использовать данный механизм в режиме **permissive**, чтобы проследить возможные ошибки в журнальных файлах.

Большое количество изменений в политике требуют повторной компиляции и сборки политики из исходного кода, однако существует возможность изменить

незначительные части политики, путем задания. Например, по умолчанию политика `targeted` не позволяет веб-серверу Apache предоставлять доступ к файлам домашних каталогов пользователей. Для того чтобы это стало возможным, значение параметра `httpd_enable_homedirs` должно быть установлено равным единице. Выбор значений множества параметров можно выполнять, используя графическую утилиту SELinux Management Tool. Альтернативный способ установки параметров политик SELinux заключается в использовании команды `setsebool -P <параметр> <значение>`. Для просмотра текущего значения параметра используется команда `getsebool <параметр>`.

3.1.2. Использование утилиты SELinux Troubleshooting Tool

По умолчанию система SELinux посылает служебные сообщения в файл `/var/log/messages`, если не используется система аудита событий. Если же она используется, то данные события записываются в файл `/var/log/audit/audit.log`. События SELinux помечаются меткой **AVC**, поэтому их легко определить среди других событий. Для анализа данных событий в ОС Linux используется утилита **SELinux Troubleshooting Tool**, для использования которой необходимо установить пакет `setroubleshoot`.

Демон `setroubleshootd`, входящий в состав данного пакета используется для запуска процесса анализа. Все данные анализа журналируются в файл `/var/log/setroubleshootd.log`. Для просмотра результатов работы анализа необходимо запустить утилиту **SELinux Troubleshooting Tool** из меню **Administration -> SELinux Troubleshooting**.

3.1.3. Работа с контекстами безопасности

Когда механизм SELinux активизирован в системе, все объекты файловой системы имеют **контекст безопасности**. Ядро разрешает доступ процессу к объекту только в том случае, если данный тип доступа разрешен политикой SELinux. Политика SELinux, в свою очередь, для управления доступом к объектам файловой системы, помечает каждый файл при помощи контекста безопасности.

Алгоритм работы механизма SELinux заключается в следующем. Если приложение пытается получить доступ к объекту, например, к файлу или сокету, то механизм SELinux проверяет в ядре кэш вектора доступа (**Access Vector Cache (AVC)**), в котором находятся кэшированные данные о правах доступа к объекту. Если решение о контроле доступа не может быть сделано на основе данных кэша AVC, то проверяется контекст безопасности приложения и самого объекта. На основании контекста безопасности приложения и самого объекта решается вопрос о предоставлении или запрете прав доступа приложения к данному объекту. После проверки, если доступ к объекту был запрещен, в журнале `/var/log/messages` будет выведено сообщение, содержащее строку **avc: denied**.

Контекст безопасности процессов, например, таких как `httpd` имеет вид `httpd_t`.

Контекст безопасности файлов называется *файловым контекстом* и записывается в расширенных атрибутах файла. Файловый контекст имеет четыре поля, разделенных символом двоеточия:

```
user:role:type:mls
```

В поле **user** указывается пользователь, который создал данный файл. Поле **role** определяет роль объекта или файла. Поле **type** определяет правило, ассоциированное с данным объектом. До тех пор пока не будет использована политика **MLS**, четвертое поле не используется. Пример файлового контекста в случае использования политики **targeted** имеет следующий вид:

```
system_u:object_r:etc_r
```

В данном примере файл идентифицируется в поле **system_u**; он является объектом, помеченным меткой **object_r**, и подчиняется правилу **etc_r**, так как данный файл содержится в каталоге **/etc**.

Для просмотра файлового контекста объектов файловой системы используется команда **ls -Z**. Результат просмотра выводится в виде, показанном в следующем листинге:

```
-rw-r--r-- root root system_u:object_r:etc_t:s0 apmd
drwxr-xr-x root root system_u:object_r:etc_t:s0 apm-scripts
-rw-r----- root root system_u:object_r:etc_t:s0 auditd
-rw-r--r-- root root system_u:object_r:etc_t:s0 authconfig
-rw-r--r-- root root system_u:object_r:etc_t:s0 autofs
-rw-r--r-- root root system_u:object_r:etc_t:s0 bluetooth
```

Для просмотра контекста безопасности процессов используется команда **ps -Z**.

В процессе копирования файлов для указания их контекста безопасности используется следующая команда:

```
cp -Z <context> file <путь_к_файлу>
```

Если ключ **-z** указан не будет, то для копируемого файла создастся новый контекст безопасности на основе процесса, который его создал, и каталога, в котором находится созданный файл. При использовании команды **mv** контекст безопасности файла сохраняется. Если необходимо изменить контекст безопасности перемещенного файла, то следует использовать следующую команду:

```
chcon <контекст> <имя_файла>
```

После того, как контекст безопасности файла будет изменен, необходимо в этом убедиться, используя команду **secon -f <имя_файла>**.

3.2. Организация межсетевого экрана (iptables)

Существует три разновидности *межсетевой экран* (МЭ) в зависимости от того, на каком уровне они осуществляют фильтрацию:

- на уровне пакетов;
- на уровне сервисов;

- с учетом состояния сессии.

Ядро ОС Linux содержит важный компонент – **netfilter**. Доступ к данному компоненту в пространстве пользователя осуществляется через утилиту **IPTables**, которая сообщает ядру, что нужно сделать с поступающими, транзитными и исходящими IP-пакетами. Наиболее часто используемыми возможностями IPTables является фильтрация пакетов и трансляция сетевых адресов (NAT). Но поскольку к ядру ОС Linux можно подключать дополнительные модули, то функционал IPTables можно существенно расширить, используя, например, возможности фильтрации пакетов на уровне L7.

Общая последовательность работы компонента **netfilter** заключается в следующем:

1. При помощи команд **iptables** пользователь сообщает ядру каким образом необходимо управлять пакетами.
2. Ядро анализирует заголовки всех пакетов, которые проходят через сетевые интерфейсы.
3. В процессе анализа заголовков ядро находит заданные соответствия, затем управление пакетом выполняется согласно правилу, заданному в команде **iptables**.

Компонент **netfilter** имеет несколько *таблиц*, в каждой из которых по умолчанию находится некоторый набор *правил*, который называется *цепочкой*. Каждое правило определяет *условие*, которому должны удовлетворять пакеты, чтобы их можно было обработать данным правилом, и *действие*, которое надо выполнить над пакетами, которые совпали с заданным условием.

В **iptables** по умолчанию доступны следующие таблицы:

- **Filter** – используется для фильтрации пакетов;
- **Nat** – используется для управления IP-адресами отправителя и получателя пакета;
- **Mangle** – используется для модификации заголовков IP-пакета;
- **Raw** – используется для управления пакетами при отслеживании сетевых соединений.

По умолчанию в ядро загружается таблица фильтрации (**filter**), которая содержит три цепочки:

- **INPUT**, содержащая правила для пакетов, в заголовке назначения которых указан данный хост, на котором настроена фильтрация пакетов;
- **FORWARD**, содержащая правила для пакетов, которые маршрутизируются на другие хосты, отличные от того, где настроена фильтрация пакетов;
- **OUTPUT**, содержащая правила, предназначенные для управления пакетами, созданными тем хостом, на котором выполняется фильтрация, и отправленных на удаленные хосты.

Если загружены дополнительные модули, такие как NAT и MANGLE, то загружаются дополнительные таблицы **nat** и **mangle**. Таблица **nat** используется для

подмены IP-адресов отправителя и получателя и содержит три цепочки:

- **PREROUTING**, в которой ядро выполняет анализ пакетов, прежде чем произвести маршрутизацию пакетов; данная цепочка используется в основном для изменения IP-адреса места назначения пакета (**DNAT**);
- **POSTROUTING**, в которой ядро выполняет анализ пакетов, после того, как произведена их маршрутизация; цепочка используется в основном для изменения IP-адреса источник, пославшего пакеты (**SNAT**);
- **OUTPUT**, применяющаяся для пакетов, которые генерируются локальным хостом, фильтруются и направляются на удаленные хосты.

Таблица **mangle** используется для модификации заголовков пакетов, например, **TOS**² и **TTL**³, до начала процесса маршрутизации или после него. Также существует возможность пометить пакеты с целью дальнейшего управления ими. Данная таблица имеет следующие цепочки:

- **PREROUTING**;
- **INPUT**;
- **FORWARD**;
- **OUTPUT**;
- **POSTROUTING**.

Для построения МЭ на основе IPTables в системе необходимо установить пакет **iptables**. Команда **iptables** имеет следующий синтаксис:

```
iptables -t <таблица> <операция> <условие> -j <действие>
```

В аргументе **-t <таблица>** указывается таблица, в которой необходимо выполнить определенную операцию. В аргументе **<операция>** указывается одна из следующих операций над цепочками:

- **-L (--list) <цепочка>** – отображение всех правил в указанной цепочке;
- **-P(--policy) <цепочка> <политика>**, которая задает политику по умолчанию для указанной цепочки; в качестве аргумента **<политика>** указывается одно из стандартных действий: **ACCEPT**, **REJECT**, **DROP**;
- **-N (--new) <цепочка>** – создание новой цепочки правил;
- **-F (--flush) <цепочка>** – удаление всех текущих правил в указанной цепочке;
- **-X (--delete-chain) <цепочка>** – удаление указанной цепочки правил;
- **-Z (--zero) <цепочка>** – сброс счетчика совпадения пакетов в указанной цепочке.

Если название цепочки не указано, то при использовании аргументов **-L**, **-F**, **-D** и **-Z**, операция выполняется для всей таблицы.

2 Модификация данного заголовка используется для организации приоритетности пакетов (**QoS**).

3 Модификация данного заголовка используется для ограничения количества хостов, которые могут принимать данные пакеты.

Помимо операций над цепочками, в качестве аргумента **<операция>** могут указываться операции над правилами в цепочке. При этом используются следующие аргументы:

- **-A(--append) <цепочка>** – добавление правила в указанную цепочку;
- **-I(--insert) <цепочка> <N>** – вставка правила в указанную цепочку в позицию **<N>**;
- **-R(--replace) <цепочка> <N>** – замена правила в позиции **<N>** в указанной цепочке;
- **-D(--delete) <цепочка> <N>** – удаление правила в позиции **<N>** в указанной цепочке.

Если порядковый номер правила, заданный в параметре **<N>** не указан, то при добавлении правила оно помещается в конец цепочки; а при вставке – в начало цепочки. Если порядковый номер не указан в аргументе **-D**, то необходимо указать полный вид правила, которое необходимо удалить.

После аргумента **<операция>** в общем синтаксисе команды **iptables** указываются аргументы **<условие>**, которые определяют соответствие пакетов. В качестве условия может использоваться большое множество образцов, которые условно можно разделить на следующие основные уровни.

L2. На данном уровне указываются сетевые интерфейсы, использующиеся для приема и передачи пакетов. Все пакеты условно можно разделить на принятые интерфейсом (**-i**) и посланные с интерфейса (**-o**). Например, запись **-i eth0** определяет все пакеты, приходящие на сетевой интерфейс **eth0**. Помимо прямого указания названия сетевого интерфейса, можно использовать символ «+», например, запись **-i eth+** определяет все пакеты, приходящие на все сетевые интерфейсы начинающиеся с имени **eth**. Кроме того, можно указывать логическое отрицание в виде символа «!», например, запись вида **-o ! eth0** определяет пакеты, которые не будут использовать данное правило, если они были отправлены с интерфейса **eth0**.

L3. На данном уровне указываются IP-адрес источника и получателя, IP-адреса подсетей в формате **X.X.X.X/NN**, а также DNS имена хостов отправителя и получателя. Например, запись **-s 172.16.0/24** определяет все пакеты, в заголовке **SOURCE** которых содержится IP-адрес из указанной подсети.

L4. На данном уровне выполняется поиск соответствий пакетов условиям, созданным на основе протоколов и портов, причем для протоколов существует возможность указывать дополнительные параметры заголовков, например, **--tcp-flags**. Для портов можно также указывать порт отправителя (**--sport <порт>**) и получателя (**--dport <порт>**). Для того, чтобы просмотреть возможные варианты опций для протоколов, можно использовать команду **iptables -p <протокол> --help**.

L7. На данном уровне возможно искать соответствие пакетов, принадлежащих конкретному приложению. Механизм по умолчанию отсутствует в

IPTables, для его использования необходимо установить в ОС Linux дополнительные модули ядра⁴.

ОС Linux имеет почти неограниченные возможности фильтрации IP-пакетов. Помимо указанных уровней соответствия может использоваться масса других условий, например, поиск текстовых данных в пакете. Более того, существует возможность разрабатывать свои собственные условия⁵ соответствия пакетов.

Все описанные условия можно группировать в единую последовательность. После того как в команде **iptables** будут указаны все условия соответствия, в аргументе **-j <действие>** необходимо указать одно из следующих действий, которое необходимо выполнить над пакетом:

- **ACCEPT**. Пакет будет принят и дальнейшие правила учитываться не будут.
- **DROP**. Пакет будет удален, без отправки сообщений о причине хосту, который послал данный пакет. Дальнейшие правила учитываться не будут.
- **REJECT**. Данное действие аналогично предыдущему с той лишь разницей, что удаленному хосту будет отправляться уведомление ICMP «**port unreachable**». Тип посылаемого уведомления может быть изменен при помощи аргумента **--reject-with**.
- **<цепочка>**. Данное действие направляет пакет для анализа в указанную цепочку. Это действие используется, если администратор создал дополнительные цепочки правил.
- **LOG**. Данное действие используется для записи информации о найденных соответствиях в журнальные файлы. Его полезно использовать для диагностики проблем в работе МЭ на основе **IPTables**.

Указанный список действий является далеко не полным, в частности, в нем не указаны действия **MARK** (используется для пометки пакетов) и **BALANCE** (используется для организации простого распределения соединений между несколькими хостами). Для просмотра всех возможных действий необходимо воспользоваться страницами руководства **man** для команды **iptables**.

В качестве примера использования таблиц **iptables** рассмотрим создание пользовательской цепочки правил, которые разрешают доступ по протоколу SSH только определенным хостам.

1. Для начала создадим новую цепочку правил с именем SSH:

```
iptables -N SSH.
```

2. Затем следует указать ядру, что все входящие пакеты необходимо обрабатывать с учетом созданной цепочки:

```
iptables -A INPUT -p tcp --dport 22 -j SSH.
```

4 Реализация фильтрации пакетов на уровне приложений представлена на сайте <http://l7-filter.sourceforge.net/>.

5 На сайте проекта Netfilter представлены дополнительные расширения условий соответствия пакетов. Одним из таких дополнений являются расширения «patch-o-matic».

3. Теперь необходимо определить хосты, которым будет разрешен доступ по протоколу SSH, добавив в цепочку SSH следующие правила:

```
iptables -A SSH -s 172.16.0.100 -j ACCEPT
iptables -A SSH -s 172.16.0.101 -j ACCEPT.
```

4. Затем для проверки результатов можно добавить правило, которое будет записывать в syslog события, удовлетворяющие созданным правилам:

```
iptables -A SSH -m limit --limit 5/s -j LOG.
```

5. Далее следует закрыть доступ по протоколу SSH для всех остальных хостов:

```
iptables -A SSH -j DROP.
```

6. Для того, чтобы просмотреть содержимое цепочки правил SSH, необходимо выполнить команду:

```
iptables -L SSH.
```

После того как будут созданы и настроены все необходимые цепочки правил **IPTables**, необходимо сохранить сделанные изменения при помощи команды **service iptables save**. Все текущие настройки будут сохранены в файле **/etc/sysconfig/iptables**. Для активации таблиц **IPTables** на этапе загрузки системы необходимо включить сервис **iptables** на соответствующем уровне выполнения, используя команды **chkconfig**. Для менее гибкой настройки МЭ можно использовать графическую программу **Red Hat Enterprise Linux firewall**, запустив команду **system-config-securitylevel** в графической оболочке.

3.3. Организация виртуальных частных сетей (VPN)

С развитием глобальной сети Интернет современные информационные системы становятся все более гибкими и масштабируемыми, используются все более и более оптимальные способы объединения распределенных источников данных. Распределенный характер деятельности большинства организаций и ценность передаваемой информации диктует соответствующие требования по безопасности к сетям передачи данных. На смену дорогим частным выделенным каналам приходят более дешевые публичные каналы, такие как глобальная сеть Интернет. В связи с этим большое значение уделяется конфиденциальности и анонимности данных. Подобные проблемы поднимаются не только при объединении распределенных площадок в единую сеть, но также для предоставления безопасного доступа к корпоративным сетевым ресурсам из публичных источников. Для решения всех поставленных вопросов была разработана технология **виртуальных частных сетей VPN** (Virtual Privat Network).

В общем случае VPN это:

- *Виртуальная сеть*: Весь обмен данными между отправителем и получателем происходит по виртуальным туннелям, не имеющим прямого сетевого доступа друг к другу.

- *Частная сеть*: Доступ к информации, передаваемой в сетях VPN, имеют только участники информационного обмена,

Общий принцип действия любой VPN сети является инкапсуляция сетевых пакетов какого-либо протокола на определенном сетевом уровне в зашифрованный туннель. В сетях TCP/IP сетевые пакеты состоят из двух частей – заголовка и данных. Заголовок определяет отправителя и получателя пакета и служебную информацию для передачи. С точки зрения сетей построенных по технологии **Ethernet**, данные пакеты называются *фреймами*. С точки зрения сети Интернет данные пакеты называются *дейтаграммами* или просто *пакетами*. Таким образом, можно сформулировать общие особенности всех сетей VPN:

- Сеть VPN инкапсулирует каждый принятый сетевой пакет, включая его заголовки и данные, в пакет *собственного формата*;
- Все части сетевого пакета шифруются;
- Сетевые пакеты изменяются: добавляется информация об используемой сети VPN и новых адресатах.

Единственное различие всех VPN сетей заключается в способе инкапсуляции и защиты данных, содержащихся в пакете. За основу работы VPN сетей взят протокол **GRE** (General Routing Encapsulation), использующийся для туннелирования данных. Все сети VPN принято различать по:

1) Назначению:

- Объединение нескольких сетей в единую виртуальную сеть (сеть-сеть).
- Подключение удаленных пользователей к сети VPN с целью получения доступа к ресурсам внутренней сети (узел-сеть).
- Передача данных между двумя отдельными хостами (узел-узел).

2) Используемому сетевому уровню (OSI) и типу протокола:

- Канальный уровень (Layer 2): На данном уровне возможна инкапсуляция любого более высокоуровневого протокола. К основным протоколам организации VPN на данном уровне являются PPTP, L2F, L2TP, L2Sec.
- Сетевой уровень (Layer 3): На данном уровне в основном используется протокол IPSec, каждый пакет которого проходит процедуру аутентификации и шифрования.
- Транспортный уровень (Layer 4): На данном уровне используются такие протоколы шифрования как TLS\SSL, SSH. В самом простом случае для туннелирования трафика используется перенаправление портов средствами SSH.

В данном пособии рассматривается ПО OpenVPN, объединяющее в себе лучшие возможности других технологий VPN. OpenVPN может использоваться на уровнях L2 и L3, выполняя при этом шифрование всего трафика по технологии SSL/TLS.

В процессе создания сетей VPN должны выполняться следующие условия:

- **Конфиденциальность:** передача данных должна быть предоставлена только для авторизованных источников;
- **Целостность:** данные должны передаваться между отправителем и получателем в неизменном виде;
- **Доступность:** передаваемые данные должны быть доступными при необходимости.

Для достижения конфиденциальности данных используется их шифрование паролями или ключами шифрования. В случае использования одинаковых ключей двумя участниками обмена, шифрование называется **симметричным**. В данном случае ключ шифрования должен присутствовать у всех участников информационного обмена. Данный способ шифрования используется в технологии IPSec: каждый ключ действителен в течении заданного промежутка времени, при этом все участники информационного обмена обмениваются ключами с тем чтобы у каждого был новый действительный ключ. Обмен ключами выполняется по собственному протоколу IKE (Internet Key Exchange).

Другой механизм шифрования заключается в использовании пары ключей каждым участником информационного обмена. Каждая пара ключей содержит два ключа – **закрытый** и **открытый** ключ. Обмену подлежит только открытый ключ, который используется для шифрования отправляемых данных. Расшифровка данных сообщений может быть выполнена только соответствующим закрытым ключом, находящимся у получателя данных. Такой способ шифрования называется **асимметричным**.

При использовании технологии SSL/TLS применительно к сетям VPN шифрование и аутентификация участников обмена выполняется при помощи **сертификатов**. **Удостоверяющий центр** сертификатов создает сертификаты и подписывает их. Каждый участник информационного обмена обязательно должен иметь действительный сертификат, выданный удостоверяющим центром для того чтобы установить связь с сетью VPN. На каждом узле может так же присутствовать список отозванных сертификатов (Certificate Revocation List (CRL)), в котором содержатся недействительные сертификаты или сертификаты клиентов, которым отказано в доступе к сети VPN. Соединение клиента с сетью VPN будет отклонено в случае если:

- Клиент не предоставил сертификат;
- Предоставленный сертификат не является сертификатом, выданным удостоверяющим центром;
- Предоставленный сертификат содержится в списке отозванных сертификатов.

Общая последовательность действий по созданию системы безопасности на основе сертификатов следующая:

1. Создание сертификата удостоверяющего центра, который будет использоваться для подписи и аннулирования сертификатов.
2. Создания ключа и запроса на подпись сертификата для клиента.

3. Подпись запроса клиента сертификатом удостоверяющего центра.
4. Распространение ключей и сертификатов между участниками информационного обмена сети VPN.

Отличительные особенности OpenVPN по сравнению с другими решениями заключаются в следующем:

- **Работа на уровнях L2 и L3.**
На втором уровне возможно туннелирование Ethernet фреймов, пакетов IPX или NETBIOS.
- **Поддержка работы через произвольный МЭ.**
Туннелирование возможно выполнять через произвольный МЭ или прокси-сервер. Имеется поддержка аутентификации при использовании совместно с прокси. На МЭ достаточно открыть всего лишь один сетевой порт для приема входящих соединений.
- **Работа в режиме сервера или клиента по протоколам TCP и UDP.**
Полностью поддерживаются протоколы TCP и UDP.
- **Поддержка работы через NAT.**
Клиент и сервер OpenVPN могут работать в частной сети, при этом туннелируемый трафик может передаваться средствами МЭ поверх публичных сетей.
- **Универсальность туннелирования.**
Туннелирование возможно для любого IP протокола, в том числе и IPSec.
- **Гибкость настройки при помощи сценариев.**
OpenVPN предоставляет много возможностей в процессе организации VPN соединения для выполнения аутентификации, отказоустойчивости и других задач проверки клиента или сервера.
- **Упрощенная сетевая архитектура.**
На сетевом уровне используется драйвер TUN/TAP, существенно упрощающий конфигурацию и повышающий безопасность использования VPN.
- **Кросс-платформенность и поддержка большого количества устройств.**
OpenVPN поддерживает достаточно большое количество устройств, в том числе Windows Mobile, Nokia Maemo, OpenWrt и прочие.
- **Наличие активного сообщества разработчиков и коммерческого варианта.**
При необходимости можно воспользоваться бесплатной поддержкой сообщества OpenVPN или приобрести за небольшую цену коммерческий вариант с дополнительными возможностями и поддержкой.

На сетевом уровне OpenVPN использует универсальный TUN/TAP драйвер. Данный драйвер позволяет ядру Linux туннелировать сетевой трафик. Со стороны пользователя и приложений он выглядит как обычный сетевой интерфейс. Устройство типа TUN может использоваться подобно виртуальному PPP интерфейсу, каким является обычный модем или DSL линк. Данный режим работы называется *маршрутизируемым*, так как в нем выполняется определение маршрутов до

участников информационного обмена в рамках сети VPN. Устройство типа TAP используется как обычный Ethernet адаптер. Весь трафик, посланный на данное устройство, шифруется и передается другим участникам информационного обмена в сети VPN. Данный режим работы называется «*мост*», поскольку он объединяет несколько разных сетей, как будто они соединены аппаратным «мостом». Таким образом, можно сформулировать следующий *принцип работы OpenVPN*:

Процесс OpenVPN ожидает поступления трафика на TUN/TAP устройствах. После поступления трафика, он шифруется и передается другому участнику в рамках сети VPN, на котором, в свою очередь, другой процесс OpenVPN принимает данные, расшифровывает их и передает приложению, ожидающему данные.

Для установки OpenVPN потребуется наличие следующих программных компонентов системы:

- Поддержка драйвера TUN/TAP⁶ на уровне ядра;
- Библиотеки OpenSSL;
- Библиотека LZ0⁷, используемая для сжатия трафика;
- Бинарный RPM пакет OpenVPN или архив с исходным кодом.

Бинарный RPM пакет доступен на сетевом репозитории **Fedora EPEL** (Extra Packages for Enterprise Linux). Для его использования необходимо установить пакет с настройками данного репозитория:

```
# rpm -ivh http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-3.noarch.rpm
```

Затем при помощи менеджера пакетов YUM можно установить OpenVPN, выполнив команду:

```
# yum install openvpn
```

Для более детальной настройки OpenVPN рекомендуется установка из исходного кода.

3.4. Аудит системных событий (auditd)

В ОС Linux начиная с версии ядра 2.6 существует возможность журналировать такие события, как доступ к файлам и выполнение системных вызовов. Процесс организации аудита системных событий ОС Linux состоит из следующих действий.

1. Установка и настройка демона аудита (**auditd**).
2. Настройка правил аудита, удовлетворяющих заданным условиям.

⁶ Последняя версия драйвера TUN/TAP доступна по адресу: <http://vtun.sourceforge.net/tun/>

⁷ Последняя версия библиотеки LZ0 доступна по адресу: <http://www.oberhumer.com/opensource/lzo/>

3. Запуск демона аудита (**auditd**).
4. Анализ полученных данных и создание отчетов аудита.

По умолчанию системный аудит отключен. Для его активации необходимо настроить запуск ядра с параметром **audit=1**. Если данный параметр присутствует, а демон **auditd** не запущен, то события журналируются в файл **/var/log/messages**.

Для того чтобы использовать аудит, в системе необходимо установить пакет **audit**. Демон **auditd** позволяет системному администратору настраивать процесс аудита, а именно:

- задавать отдельный файл для журналирования событий;
- определять ротацию журнального файла событий;
- задавать оповещения в случае переполнения журнала событий;
- определять уровень детализации событий;
- настраивать правила аудита.

В процессе работы демон **auditd** использует конфигурационный файл **/etc/audit/auditd.conf**. На каждой строке данного файла указываются соответствующие директивы, значения которых указываются после знака равенства.

```
log_file = /var/log/audit/audit.log
log_format = RAW
priority_boost = 3
flush = INCREMENTAL
freq = 20
num_logs = 4
dispatcher = /sbin/audispd
disp_qos = lossy
max_log_file = 5
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
```

В директиве **log_file** указывается файл, куда будут журналироваться события. По умолчанию, таким файлом является файл **/var/log/audit/audit.log**. В директиве **log_format** указывается формат данных, которые необходимо журналировать. Если в данной директиве указано значение **RAW**, события записываются в файл в том виде, в котором они поступают из ядра. В директиве **flush** определяется частота журналирования событий. Если для данной директивы указано значение **INCREMENTAL**, то очистка журнала определяется на основе директивы **freq**, в которой указывается количество записей, которые принимает демон **auditd**, прежде чем записать их в журнал.

Если в файле **auditd.conf** указана директива **max_log_file_action** со значением **ROTATE**, то журнальные файлы будут ротироваться по достижении размера, заданного в директиве **max_log_file** (в Мб). В директиве **num_logs** определяется

количество журнальных файлов, которые должны быть сохранены, прежде чем будет выполнена ротация данных.

В директиве **dispatcher** задается программа, которая обрабатывает все события аудита. Она может использоваться для формирования отчетов или конвертирования журнала в формат, понимаемый другими программами анализа журнальных файлов. Директива **disp_qos** определяет параметры взаимодействия данной программы с демоном **auditd**. Если в данной директиве указано значение **lossy**, то события, посланные данной программе, удаляются, в случае переполнения буфера обмена данными между демоном **auditd** и данной программой (по умолчанию размер данного буфера составляет 128 Кб). Запись событий по-прежнему осуществляется, если в директиве **log_format** не указано значение **nolog**.

В директиве **space_left** задается предел свободного дискового пространства раздела, на котором располагаются данные аудита, а в директиве **space_left_action** – действие, которое нужно выполнить, если указанный предел превышен. Если указано значение **SYSLOG**, то в журнал сервиса **syslog** будет записано соответствующее предупреждение. В директиве **disk_full_action** указывается действие, выполняемое в том случае, если в разделе, содержащем журнальные файлы аудита дисковое пространство заполнено, а в директиве **disk_error_action** указывается действие, выполняемое при обнаружении ошибки записи данных в журнальный файл или его ротации. По умолчанию для данных директив задано значение **SUSPEND**, при котором следующие события записываются в журнальный файл аудита не будут, а действия, выполняемые в настоящий момент над журнальным файлом, завершатся.

Для настройки правил аудита системных вызовов и слежения за файлами и каталогами используется утилита **auditctl**. Если сервис **auditd** настроен на автозапуск, то правила аудита системных вызовов и слежения за объектами файловой системы можно задать в файле **/etc/audit/audit.rules**. Каждое правило аудита должно задаваться в отдельной строке. Форма записей правил и средств имеет вид аргументов командной строки для команды **auditctl**. Демон **auditd** считывает правила сверху вниз, и если будет обнаружено два конфликтующих правила, то предпочтение будет отдано первому найденному правилу.

Каждая строка описания правила аудита системных вызовов имеет следующий синтаксис:

```
-a <список>,<действие> <опции>
```

В качестве параметра **<список>** указывается список событий, в который добавляется данное правило. Это могут быть списки:

- **task** – используется для ведения событий, связанных с созданием процессов;
- **user** – используется для ведения событий, в которых указываются параметры пользовательского пространства, такие как **uid**, **pid** и **gid**;
- **exclude** – используется для запрета аудита указанных в данном списке событий;

- **entry** – используется для регистрации событий системных вызовов;
- **exit** – используется для регистрации событий системных вызовов.

В качестве параметра **<действие>** доступно всего два действия: **never** и **always**. При указании первого из них события не записываются в журнал аудита. Указание действия **always** приведет к противоположному результату.

В параметре **<опции>** задаются опции, являющиеся фильтрами, при помощи которых можно детализировать события аудита. Полный список опций можно посмотреть в руководстве **man** для команды **auditdctl**. В качестве опций можно задавать уникальные идентификаторы пользователя или процесса, название системного вызова и многое другое. При этом возможно использовать логические выражения для формирования комбинированных опций.

Добавлять правила в данный файл можно также при помощи команды **auditdctl**, используя ключ **-a**. При этом правила добавляются в конец файла **auditd.conf**. Для того, чтобы добавить правило в начало данного файла, необходимо использовать ключ **-A**.

```
-a exit,always -S open -F uid=502 -F key=502open
-a entry,always -S chown
```

В первом случае задается правило, которое фиксирует все события, связанные с системным вызовом **open**, выполняемым от пользователя с идентификатором **502**. Записи, удовлетворяющие данному условию, помечаются в файле **audit.log** меткой **502open**. Во втором случае задается правило, которое фиксирует все системные вызовы **chown**.

Когда заданное в правиле действие будет выполнено, результат его выполнения заносится в журнал **/var/log/audit/audit.log**.

```
type=PATH msg=audit(1233260763.459:334): item=0 name="/etc/passwd" inode=852389
dev=08:02 mode=0100644 ouid=0 ogid=0 rdev=00:00
type=SYSCALL msg=audit(1233260763.460:335): arch=40000003 syscall=5 success=yes
exit=3 a0=928e350 a1=8000 a2=0 a3=8000 items=1 ppid=3529 pid=3551 auid=0 uid=502
gid=502 euid=502 suid=502 fsuid=502 egid=502 sgid=502 fsgid=502 tty=pts1
comm="vim" exe="/usr/bin/vim" key="502open"
```

Помимо слежения за системными вызовами, система аудита ОС Linux позволяет следить за доступом к файлам и каталогам. Для этого в файле **audit.rules** необходимо создать правило, имеющее следующий синтаксис:

```
-w <файл> -k <ключ>
```

В параметре **<файл>** указывается полный путь к файлу или каталогу, за которым необходимо следить. Параметр **-k <ключ>** используется для пометки всех событий, связанных с доступом указанному файлу. Это позволяет быстро находить интересующую информацию в журнальном файле. В следующем примере приведены два правила, в которых осуществляется слежение за файлами в каталоге **/etc/sysconfig**, а также слежение за доступом к команде **iptables**:

```
-w /etc/sysconfig -k SYSCONFIG
-w /sbin/iptables -k IPTABLES -p x
```

В ключах **-k** указываются соответствующие метки событий. Ключ **-p x** во вто-

ром правиле указанного примера обозначает, что необходимо рассматривать только события, связанные с выполнением данного файла.

Для управления правилами, помимо редактирования конфигурационного файла **audit.rules**, можно использовать команду **auditctl**. Для просмотра активных правил аудита необходимо выполнить команду **auditctl -l**. Для определения статуса системы аудита необходимо выполнить команду **auditctl -s**.

Обычно журнальный файл системы аудита событий ОС Linux можно просматривать при помощи любого текстового редактора. Однако для выполнения общего анализа событий требуется суммировать отдельные записи в данном файле, что практически нереально выполнить, используя только возможности текстового редактора. Для выполнения анализа и отображения статистики аудита событий в состав пакета **audit** входят утилиты **ausearch** и **aureport**. Утилита **ausearch** может выполнять поиск записей аудита на основе различных критериев поиска, таких как имя файла, идентификатор пользователя, название системного вызова и прочее. Утилита **aureport** используется для формирования отчетов на основе журнального файла аудита **audit.log**.

Для формирования более детального отчета необходимо выполнять команду **aureport** с использованием дополнительных ключей, которые можно найти в руководстве **man** по данной команде. Например, для отображения статистики об измененных файлах, необходимо запустить команду **aureport** с ключом **-c**.

Система аудита позволяет следить за выполнением определенного процесса, используя утилиту **autrace <команда>**. Это особенно полезно при поиске ошибок в работе различных демонов.



Приложения

Приложение 1. Конфигурационный файл загрузчика GRUB2

Параметр	Значения	Описание
GRUB_DEFAULT	Целое число (начиная с 0) или « saved »	Определяет пункт меню для загрузки по умолчанию (число). Определяет пункт меню по умолчанию при помощи команды grub-set-default X , где X – номер пункта меню.
GRUB_TIMEOUT	Целое число (сек.) или 0	Определяет время задержки перед загрузкой пункта меню по умолчанию.
GRUB_HIDDEN_TIMEOUT	Целое число (сек.) или 0	В случае указания целого числа больше 0, меню загрузки не отображается и через указанное время выполняется загрузка пункта меню по умолчанию. В случае указания 0 и нажатия произвольной клавиши будет отображено меню загрузчика.
GRUB_HIDDEN_TIMEOUT_QUIET	true или false	True (false) – не отображать (отображать) обратный отсчет времени.
GRUB_CMDLINE_LINUX	“параметр”	Добавляет указанные параметры к пункту меню загрузки ядра linux,
GRUB_CMDLINE_LINUX_DEFAULT	“параметр”	Задаёт параметры загрузки ядра в штаном режиме. При использовании параметра splash отображается дополнительная информация.
GRUB_TERMINAL	console	Отключает загрузку графического меню GRUB.

Параметр	Значения	Описание
GRUB_DISABLE_LINUX_UUID	true	Не передавать ядру параметр root=UUID=xxx .
GRUB_GFXMODE	M x N	Задаёт разрешение графического меню GRUB.
GRUB_DISABLE_LINUX_RECOVERY	true	Не отображать в меню загрузчика пункт восстановления.
GRUB_DISABLE_OS_PROBER	true	Запрещает использования программы поиска операционных систем на дисковых разделах.

Приложение 2. Описание основных категорий и групп пакетов ОС Linux

Категория пакетов	Группы пакетов	Описание
Desktop Environments (Рабочие окружения)	GNOME Desktop Environment	Графическая среда GNOME; используется в ОС Linux по умолчанию
	KDE Desktop Environment	Графическая среда KDE
Applications (Приложения)	Authoring and Publishing	Средства работы с документами, такие как DocBook, LinuxDoc и TeX
	Editors	Текстовые редакторы vi и emacs
	Engineering and Scientific	Приложения, предназначенные для математических вычислений и научного использования
	Games and Entertainment	Игры и развлекающие программы
	Graphical Internet	Графические средства работы с Интернетом – Интернет-обозреватели, FTP-клиенты
	Graphics	Средства для редактирования изображений, включающие редактор GIMP
	Office/Productivity	Офисные программы – OpenOffice, просмотрщик файлов формата PDF и др.

Категория пакетов	Группы пакетов	Описание
	Sound and Video	Проигрователи аудио- и видеофайлов
	Text-based Internet	Консольные Интернет-обозреватели, такие как elinks, lynx
Development (Разработка)	Development Libraries	Библиотеки, необходимые для разработки приложений
	Development Tools	Программы для разработки приложений, такие как компилятор gcc, интерпретаторы perl и python, утилита make
	GNOME Software Development	Программы, предназначенные для разработки элементов среды GNOME
	Java Development	Средства разработки Java-приложений
	KDE Software Development	Программы, предназначенные для разработки элементов среды KDE
	Legacy Software Development	Устаревшие библиотеки
	Ruby	Язык программирования Ruby
	X Software Development	Программы, предназначенные для разработки приложений системы X Window
Servers (Серверные компоненты)	DNS Name Server	Сервер DNS, основанный на Berkeley Internet Name Domain (BIND)
	FTP Server	Сервер FTP, основанный на Very Secure FTP Daemon (vsftpd)
	Legacy Network Servers	Устаревшие сетевые службы – rsh, telnet, tftp
	Mail Server	Сервер почтовых сообщений (sendmail или postfix) и программы доступа к почтовым ящикам по протоколу IMAP
	MySQL Database	База данных MySQL
	News Serve	Сервер новостей Internet Network News (INN)
	Network Servers	Сетевые сервисы DHCP и NIS
	PostgreSQL Database	База данных PostgreSQL
	Printing Support	Поддержка печати CUPS

Категория пакетов	Группы пакетов	Описание
	Server Configuration Tools	Графические утилиты сетевого администрирования
	Web Server	Веб-сервер Apache и прокси-сервер Squid, а также соответствующие им средства администрирования
	Windows File Server	Сервер Samba, позволяющий управлять доступом windows клиентов, работающих в среде Windows
Base System (Базовая система)	Administration Tools	Графические утилиты системного администрирования
	Base	Минимальный набор программ, необходимый для работы ОС Linux
	Dialup Networking Support	Средства работы со связью, использующие протоколы PPP, PPPoE, ISDN, DSL
	Java	Язык программирования Java
	Legacy Software Support	Средства для совместимости со старыми версиями программного обеспечения
	System Tools	Различные системные утилиты
	X Window System	Система X Window
Languages (Языки)	Language Support	Поддержка различных языков

Приложение 3. Основные команды управления LVM

Команда	Описание
Команды управления физическими томами	
pvchange	Смена атрибутов физического тома, например, запрет использования физических экстендов отдельных дисков
pvcreate	Инициализация диска или раздела как физического тома
pvdisplay	Отображение настроенных физических томов

Команда	Описание
pvmove	Перенос физических томов с указанного раздела в свободное место другого раздела
pvremove	Отмена инициализации диска или раздела как физического тома
pvresize	Увеличение размера физического тома
pvs	Отображение настроенных физических томов и соответствующих им групп томов
pvscan	Отображение настроенных физических томов и соответствующих им групп томов
Команды управления группами томов	
vgcfgbackup vgcfgrestore	Резервирование и восстановление конфигурационных файлов LVM, находящихся в каталоге /etc/lvm
vgchange	Активация или деактивация группы томов
vgconvert	Конвертация группы томов из версии LVM1 в LVM2
vgcreate	Создание группы томов из двух и более физических томов
vgdisplay	Отображение настроек текущих групп томов
vgexport vgimport	Используется для экспорта и последующего импорта физических томов на другую систему
vgextend	Расширение размера группы томов за счет дополнительно созданных физических томов
vgmerge	Слияние двух групп томов
vgmknodes	Разрешение проблем с устройствами групп томов
vgreduce	Уменьшение размера группы томов за счет исключения указанного физического тома из данной группы
vgremove	Деактивация группы томов. После выполнения команды, группа томов не будет соотнесена ни с одним логическим томом
vgrename	Переименование группы томов
vgs	Отображение базовой информации о настроенных группах томов
vgscan	Сканирование созданных групп томов и отображения информации о них
Команды управления логическими томами	
lvchange	Смена атрибутов логического тома

Команда	Описание
lvconvert	Конвертация логических томов из последовательного режима работы в дублирующий; создание мгновенных снимков
lvcreate	Создание нового логического тома в группе томов
lvdisplay	Отображение текущих настроенных логических томов
lvextend	Увеличение размера логического тома
lvreduce	Уменьшение размера логического тома
lvremove	Удаление размера логического тома
lvrename	Переименование логического тома
lvresize	Изменение размера логического тома
lvs	Отображение всех настроенных логических томов
lvscan	Сканирование и отображение всех настроенных логических томов

Приложение 4. Основные журнальные файлы ОС Linux

Журнальный файл	Краткое содержание
acpid	Сообщения демона acpid, отражающие работу ACPI
anaconda.*	Сообщения программы anaconda: anaconda.log – сообщения инсталляции; anaconda.syslog – первый вывод утилиты dmesg; anaconda.xlog – сообщения о запуске сервера X Window
audit/	Сообщения системы аудита ядра ОС Linux
boot.log	Сообщения о запуске и останове сервисов в процессе загрузки (или завершения работы) ОС Linux
btmpt	Сообщения о попытках неудачной аутентификации в системе
cron	Сообщения планировщика задач cron
cups/	Сообщения печати и доступа к принтерам
dmesg	Сообщения начальной загрузки
gdm/	Сообщения связанные с запуском среды GNOME

Журнальный файл	Краткое содержание
kdm	Сообщения связанные с запуском среды KDE
httpd/	Журнальные файлы веб-сервера Apache
lastlog	Записи входа пользователей в систему
mail/	Журнальные файлы статистических почтовых данных
maillog	Журнальные файлы почтовых сервисов Sendmail или Postfix
messages	Сообщения всех остальных средств, описанных в файле syslog.conf
prelink.log	Сообщения о библиотеках и бинарных файлах, ускоряющих процесс загрузки ОС Linux
rpm_pkgs	Список установленных пакетов
samba/	Журнальные файлы сервиса Samba
scrollkeeper.log	Сообщения связанные с документацией GNOME
secure	События аутентификации и доступа к системе с удаленных хостов
setroubleshoot	Сообщения связанные с утилитой SELinux troubleshooting tool
spooler	Сообщения очереди печати
squid/	Журнальные файлы прокси-сервера Squid
wtm	Сообщения входа в систему в бинарном виде. Просматривается утилитой utmpdump
Xorg.0.log	Сообщения о работе системы X Window
yum.log	Сообщения связанные с менеджерами пакетов yum или его графическим аналогом – rpm



Практические работы

Описание виртуальных машин

DNS-имя	IP-адрес	Характеристики
srv.linux.lab	172.16.0.2 (Host-only); 192.168.1.* (Bridge).	Виртуальная машина слушателя. ОС RHEL 5
client.linux.lab	172.16.0.3 (Host-only); 192.168.1.* (Bridge).	Виртуальная машина слушателя. ОС RHEL 5
main.linux.lab	192.168.1.110	Виртуальная машина преподавателя, являющаяся локальным репозиторием пакетов ОС Linux. Доступ к данным осуществляется по FTP под учетной записью: Логин: user Пароль: P@ssw0rd

Практическая работа 1.

Системное администрирование ОС Linux

Цель

В данной работе вам предстоит выполнить сетевую инсталляцию ОС Linux, установить ядро ОС Linux, настроить отказоустойчивое и масштабируемое хранилище данных, выполнить мониторинг системных ресурсов. Кроме того, потребуется организовать систему резервного копирования пользовательских данных и централизованную систему сбора журнальных файлов.

Предварительные требования

Для выполнения данного упражнения понадобится две виртуальных машины: **srv.linux.lab** и **client.linux.lab**. Первые два упражнения выполняются на виртуальной машине **srv.linux.lab**. В процессе выполнения четвертого, шестого и восьмого упражнений понадобится доступ к виртуальной машине преподавателя для установки необходимых пакетов и копирования файлов.

Упражнение 1.1. Изменение параметров загрузки ядра ОС Linux

Сценарий: Вам необходимо узнать информацию о том, как система определила геометрию вашего жесткого диска в процессе загрузки.

Для этого, используя меню загрузчика GRUB, вам необходимо изменить параметр ядра ОС Linux для вывода более детальной информации в процессе загрузки системы. После того, как система полностью загрузится, вам необходимо найти интересующую вас информацию в файле `/var/log/dmesg`, а затем зафиксировать измененные параметры запуска ядра ОС Linux, отредактировав соответствующий конфигурационный файл загрузчика GRUB.

Задания	Описание действий
1. Используя меню загрузчика GRUB, изменить параметр ядра ОС Linux, отвечающий за детализацию информации, отображаемой при загрузке системы.	1. Находясь в режиме командной строки, выполнить перезапуск системы. 2. Дождаться появления экрана меню загрузчика GRUB, нажав на клавишу <Space> в процессе обратного отсчета времени. 3. В появившемся меню ввести команду « a » для изменения параметров загрузки ядра. 4. Удалить строку quiet и нажать на клавишу <Enter> для сохранения сделанных изменений и запуска процесса загрузки ОС Linux.

Задания	Описание действий
2. Найти информацию по геометрии диска в файле /var/log/dmesg .	1. Открыть файл /var/log/dmesg в редакторе vim и найти информацию, относящуюся к устройству sda : Из файла /var/log/dmesg видно, что устройство sda представляет собой жесткий SCSI диск FAST-40 WIDE SCSI 80.0 MB/s ST, имеющий 16777216 секторов общим объемом 8590 Мбайт. 2. Определить аналогичную информацию, используя команду dmesg .
3. Изменить параметры загрузки ядра ОС Linux.	1. Удалить в конфигурационном файле загрузчика GRUB параметр ядра quiet , используя редактор vim .

Упражнение 1.2. Управление системными сервисами

Сценарий: В данной работе необходимо выполнить операции по управлению сервисом NFS, используя при этом утилиты **chkconfig** и **ntsysv**.

Задания	Описание действий
1. Настроить автозапуск сервиса NFS на третьем уровне выполнения	1. Выполнить вход в систему под пользователем root : login: root password: P@ssw0rd 2. Открыть сеанс командной строки, запустив программу « Terminal » 3. Настроить запуск сервиса NFS на третьем уровне выполнения и проверить успешность настройки, используя утилиту chkconfig .
2. Изменить уровень выполнения, используемый по умолчанию	1. Изменить уровень выполнения, используемый по умолчанию, отредактировав конфигурационный файл /etc/inittab при помощи редактора vim .
3. Выполнить перезапуск системы и проверить статус процессов сервиса NFS.	1. Выполнить перезапуск системы и дождаться приглашения команды login и войти в систему под пользователем root . 2. Проверить статус процессов сервиса NFS. 3. Определить зависимость процессов сервиса NFS, их идентификаторы PID и PPID.

Задания	Описание действий
4. Вернуться к исходной конфигурации системы, изменив параметры автозапуска сервиса NFS и уровень выполнения	1. Изменить параметры запуска сервиса NFS, используя утилиту ntsysv , и проверить сделанные изменения. 2. В появившемся окне программы ntsysv необходимо снять отметку с пункта « nfs » и выйти, нажав на кнопку « OK ». 3. Изменить уровень выполнения по умолчанию, отредактировав конфигурационный файл /etc/inittab при помощи редактора vim так же как это было сделано в первом пункте.
5. Вернуться на прежний уровень выполнения.	1. Вернуться на прежний уровень выполнения, используя команду init . 2. Дождаться появления графического экрана входа в систему, осуществить вход под пользователем root , запустить программу « Terminal » и проверить текущий уровень выполнения, используя команду who . Сделать соответствующий вывод: _____ _____ _____

Упражнение 1.3. Сетевая инсталляция ОС Linux с использованием сервиса NFS

Сценарий: Вы являетесь системным администратором веб-серверов в крупном датацентре. В связи с увеличением количества внешних клиентов и нагрузки на серверы, вы приобрели дополнительный сервер, который в данный момент находится на территориально удаленном складе, имеющим хороший Ethernet канал связи с вашим датацентром. В сети датацентра у вас имеется готовый к работе инсталляционный сервер, который содержит необходимый вам дистрибутив ОС Linux, доступный по NFS. Перед вами стоит задача как можно скорее установить ОС Linux на сервер, находящийся на удаленном складе.

Задания	Описание действий
1. Настроить на виртуальной машине srv.linux.lab инсталляционный ресурс и сделать его доступным по NFS.	1. Создать каталог /inst , в котором будет монтироваться ISO образ дистрибутива ОС Linux. 2. Смонтировать ISO образ дистрибутива ОС Linux (из папки D:\VPCImages\ISO) в точку монтирования /inst . 3. Добавить каталог /inst в список экспортируемых файловых систем в файле /etc/exports : <code>/inst *(ro,sync)</code>

Задания	Описание действий
	<p>4. Экспортировать каталог <code>/inst</code> через NFS и перезапустить сервис <code>nfs</code>:</p> <pre># exportfs -a # service nfs on # service nfs restart</pre> <p>5. Выполнить проверку экспорта содержимого каталога <code>/inst</code>, используя команду <code>showmount -e</code>.</p>
<p>2. Выполнить загрузку виртуальной машины <code>client.linux.lab</code> с загрузочного диска и начать сетевую инсталляцию.</p>	<p>1. Подключить к виртуальной машине <code>client.linux.lab</code> ISO образ <code>boot.iso</code> (из папки <code>D:\VPCImages\ISO</code>), используя консоль VMWare Player, и убедиться, что устройство CD/DVD drive подключено.</p> <p>2. В самом начале загрузки виртуальной машины <code>client.linux.lab</code> нажать клавишу <code><F2></code>. В настройках BIOS перейти на вкладку «Boot» и первым пунктом установить значение «CD-ROM Drive». Сохранить сделанные изменения и выйти <code><F10></code>.</p> <p>3. Дождаться появления загрузочного экрана ОС Linux и в приглашении <code>boot:</code> ввести команду <code>linux askmehod</code>.</p> <p>4. В окне выбора метода инсталляции выбрать пункт NFS image.</p>
<p>3. Указать параметры расположения инсталляционного ресурса</p>	<p>1. В поле «NFS server name» указать IP-адрес сервера NFS.</p> <p>2. В поле «Red Hat Enterprise Linux Server directory» указать расположение каталога с инсталляционными файлами (<code>/inst</code>).</p> <p>3. В окне выбора типа инсталляции указать пункт установки ОС Linux («Install Red Hat Enterprise Linux Server») и нажать на кнопку Next.</p>
<p>4. Разметить дисковое пространство, используя конфигурацию по умолчанию.</p>	<p>1. В окне разметки дисков указать пункт «Remove all partitions on selected drives and create default layout», отметить диск <code>sda</code> и пункт «Review and modify partitioning layout» и подтвердить сделанные изменения, нажав на кнопку Next.</p> <p>В окне разметки дискового пространства, указаны разделы, которые будут форматироваться, из них:</p> <ul style="list-style-type: none"> • раздел <code>/boot</code> имеет размер 100Мбайт и тип файловой системы <code>ext3</code>. Кроме того, данный раздел является первичным разделом и отмечен как «загрузочный»; • раздел <code>swap</code> имеет размер равный удвоенному количеству оперативной памяти и тип файловой системы Linux swap; • раздел <code>/</code> создан на логическом томе LVM и занимает весь оставшийся объем свободного дискового пространства диска <code>sda</code>.

Задания	Описание действий
5. Указать место установки загрузчика GRUB и его параметры.	<p>1. Указать, что загрузчик GRUB будет установлен на устройство sda, отметив пункт «The GRUB boot loader will be installed on /dev/sda».</p> <p>2. В пункте выбора загружаемой операционной системы отметить ОС Red Hat Enterprise Linux Server. Затем нажать на кнопку Next.</p>
6. Задать параметры установки, установить необходимые пакеты и перезагрузить систему.	<p>1. В окне настроек сетевого интерфейса указать ручной выбор IP-адреса, ввести IP-адрес (172.16.0.3), имя хоста (client.linux.lab), шлюз по умолчанию (172.16.0.2), IP-адрес DNS сервера (172.16.0.2) и установить отметку «Active on boot» напротив интерфейса eth0. Затем нажать на кнопку Next.</p> <p>2. В окне настройки часового пояса отметить пункт «System clock uses UTC» и выбрать часовой пояс GMT +03. Затем нажать на кнопку Next.</p> <p>3. В качестве пароля пользователя root в двух полях ввести строку P@ssw0rd и нажать на кнопку Next.</p> <p>4. В окне выборе метода инсталляции пакетов снять отметки с пунктов Web Server и Software Development и нажать на кнопку Next. Тем самым вы подтвердите установку пакетов программ принятых по умолчанию.</p> <p>5. После завершения установки и перезагрузки системы, изменить в настройках BIOS основной виртуальной машины порядок загрузки ОС с устройства Hard Drive.</p> <p>6. Во втором окне программы Red Hat Setup Agent принять лицензионное соглашение, отметив пункт «Yes, I agree to the License Agreement».</p>
7. Выполнить постинсталляционные настройки.	<p>1. В окне настройки МЭ (Firewall) выбрать пункт Firewall: Disabled и нажать на кнопку Next.</p> <p>2. В окне настройки механизма SELinux выбрать пункт SELinux: Disabled и нажать на кнопку Next.</p> <p>3. В окне настройки утилиты Kdump отметить пункт Enable kdump?, оставить все остальные значения по умолчанию и нажать на кнопку Next.</p> <p>4. В окне настройки даты и времени (Date and Time) установить текущее время и дату и нажать на кнопку Next.</p> <p>5. В окне настройки параметров обновлений (Setup Software Updates) отметить пункт «No, I prefer to register art later time» и нажать на кнопку Next. Далее подтвердить, что подключение к сети RNH не планируется в настоящий момент, нажав на кнопку «No, thanks I'll connect later»</p>
8. Создать дополнительных пользователей.	<p>1. В окне создания дополнительных пользователей (Create User) создать пользователя test с паролем q1q1q1. Затем завершить инсталляцию.</p>

Задания	Описание действий
9. Определить количество установленных пакетов при помощи журнала <code>/root/install.log</code>	<ol style="list-style-type: none"> 1. Дождаться появления графического экрана входа в систему и ввести учетные данные пользователя root. 2. Открыть сеанс командной строки, запустив программу «Terminal» 3. Определить количество установленных пакетов при помощи журнала <code>/root/install.log</code>.

Упражнение 1.4. Установка ядра ОС Linux

Сценарий: В компании, в которой вы работаете, имеется веб-сервер, который предоставляет доступ к сайту вашей компании внешним клиентам. Для повышения надежности дисковой подсистемы данного сервера, вы приобрели дополнительный контроллер жестких дисков. После установки данного контроллера в сервер, операционная система не обнаружила установленное вами устройство. Вы созвонились с поставщиком оборудования и узнали, что для корректной работы приобретенного вами контроллера, вам необходимо обновить ядро ОС Linux до указанной версии. В целях безопасности, вы решили не обновлять существующее ядро ОС Linux, а установить новую версию ядра ОС Linux дополнительно. Данное упражнение выполняется на виртуальной машине **client.linux.lab**.

Задания	Описание действий
1. Настроить программу yum на использование сетевого репозитория	<ol style="list-style-type: none"> 1. Загрузить по FTP с виртуальной машины преподавателя конфигурационный файл rhel-base.repo и поместить его в каталог <code>/etc/yum/repos.d</code>. 2. Проверить работу менеджера пакетов yum, выполнив команду отображения группы пакетов «Web Server». В результате выполнения команды должен отобразиться список пакетов, входящих в данную группу.
2. Определить дополнительную информацию об установленном ядре и доступном пакете обновления.	<ol style="list-style-type: none"> 1. Определить дополнительную информацию об установленном ядре ОС Linux, используя менеджер пакетов rpm: <pre># rpm -qa grep kernel kernel-2.6.18-92.e15 kernel-headers-2.6.18-92.1.18.e15</pre> Пакет, содержащий ядро ОС Linux имеет название kernel-2.6.18-92.e15. <pre># rpm -qi kernel-2.6.18-92.e15</pre> В системе установлено ядро версии _____, релиз _____. 2. Определить файл, в котором содержатся параметры конфигурации ядра: <pre># rpm -ql kernel-2.6.18-92.e15 grep conf</pre>

Задания	Описание действий
	<p>Параметры конфигурации ядра содержатся в файле _____</p> <p>3. Используя менеджер пакетов yum, определить доступный пакет обновления ядра ОС Linux:</p> <pre># yum check-update</pre> <p>4. Используя менеджер пакетов yum, определить дополнительную информацию о ядре, содержащемся в обновленном пакете:</p> <pre># yum info kernel.i686</pre> <p>Обновленный пакет содержит ядро версии _____, релиз _____.</p>
<p>3. Установить обновленный пакет, содержащую обновленную версию ядра ОС Linux.</p>	<ol style="list-style-type: none"> 1. Создать список файлов, присутствующих в каталоге /boot: <pre># ls -l /boot/ > boot.list</pre> 2. Создать резервную копию конфигурационного файла загрузчика ядра GRUB: <pre># cp /boot/grub/grub.conf /boot/grub/grub.conf.old</pre> 3. Установить пакет, содержащий обновленную версию ядра ОС Linux: <pre># yum install kernel</pre> 4. Создать список файлов, присутствующих в каталоге /boot: <pre># ls -l /boot/ > bootnew.list</pre> 5. Определить файлы, которые добавились в каталог /boot после установки нового ядра: <pre># diff boot.list bootnew.list</pre>
<p>4. Запустить систему, используя обновленную версию ядра.</p>	<ol style="list-style-type: none"> 1. Выполнить перезагрузку системы: 2. В процессе перезапуска системы в меню загрузчика GRUB выбрать новую версию используемого ядра. В данном случае, загрузка нового ядра осуществляется при выборе пункта Red Hat Enterprise Linux Server (2.6.18-53.1.21.el5). 3. Выполнить вход в систему под пользователем root и просмотреть текущую версию ядра ОС Linux при помощи команды uname.

Упражнение 1.5. Создание отказоустойчивого и масштабируемого хранилища

Сценарий: Вы организовали сервис FTP, который является частью хостинговой системы, предоставляющей сервис FTP для двух заказчиков. В данной работе вам предстоит создать два раздела (**/pub1** и **/pub2**), которые будут использоваться для доступа по FTP вашими заказчиками. У каждого заказчика имеются свои требования по отказоустойчивости и масштабируемости данных, хранящихся на

вашем FTP сервере. Раздел **/pub1** должен быть создан на основе массива RAID5, организованного из 3-х дисков размером 100Мбайт каждый. Раздел **/pub2** должен быть создан на основе массива RAID1, организованного из 2-х дисков размером 100Мбайт каждый. Второй раздел должен обеспечить масштабируемость данных в пределах 100Мб. Перед вами стоит задача создать необходимые разделы и убедиться в их отказоустойчивости. Данное упражнение выполняется на виртуальной машине **srv.linux.lab**.

Задания	Описание действий
1. Создать необходимые разделы для создания RAID массивов.	1. Определить физические диски, доступные для формирования двух RAID массивов, используя команду fdisk . 2. Создать пять разделов типа linux raid (fd).
2. Создать массив RAID 5 из трех дисков и убедиться в корректности сделанных действий.	1. Создать массив RAID 5 из трех дисков: 2. Проверить корректность сделанных изменений. При правильном создании массива в файле /var/log/messages должно появиться сообщение: <pre>Jan 15 06:21:37 rhel5 kernel: RAID5 conf printout: Jan 15 06:21:37 rhel5 kernel: --- rd:3 wd:3 fd:0 Jan 15 06:21:37 rhel5 kernel: disk 0, o:1, dev:sdb1 Jan 15 06:21:37 rhel5 kernel: disk 1, o:1, dev:sdcl Jan 15 06:21:37 rhel5 kernel: disk 2, o:1, dev:sdd1</pre>
3. Создать массив RAID 1 из двух оставшихся дисков и убедиться в корректности сделанных действий.	1. Создать дополнительное блочное устройство md1 : <pre># mknod -m 640 /dev/md1 b 9 1 # chown root:disk /dev/md1</pre> 2. Создать массив RAID 1 из трех дисков: 3. Проверить корректность сделанных изменений по файлу /var/log/messages .
4. Сохранить сделанные изменения в конфигурационном файле mdadm и настроить активацию RAID массивов в процессе загрузки системы.	1. Создать конфигурационный файл /etc/mdadm.conf , который будет содержать конфигурацию RAID массивов: <pre># echo DEVICE /dev/sd[bcdefg]1 > /etc/mdadm.conf # mdadm --detail --scan >>/etc/mdadm.conf</pre> 2. Отредактировать созданный конфигурационный файл /etc/mdadm.conf следующим образом: В каждой директиве ARRAY добавить параметр devices=/dev/sdb1,/dev/sdd1,... , где через запятую указать устройства, которые принадлежат данному RAID массиву. В директиве ARRAY /dev/md1 добавить в конце строки параметр auto=yes для автоматического создания устройства. В конец файла /etc/mdadm.conf добавить две строки CREATE owner=root group=disk mode=0640 auto=yes и MAILADDR root@localhost . В результате файл должен иметь следующий вид:

Задания	Описание действий
	<pre> 1 DEVICE /dev/sd[bcdefg]1 2 ARRAY /dev/md0 level=raid5 num-devices=3 UUID=1b0c8fcf:77680e80:bb5d41b2:649c7ebb devices=/dev/ sdb1,/dev/sdc1,/dev/sdd1 3 ARRAY /dev/md1 level=raid1 num-devices=3 UUID=3cb2 b9b2:4b4e53b3:0d7670b9:552e8902 devices=/dev/sde1,/dev/ sdf1,/dev/sdg1 auto=yes 4 CREATE owner=root group=disk mode=0640 auto=yes 5 MAILADDR root@localhost </pre> <p>Здесь и далее по тексту в листингах файлов цифрами отмечены номера строк файла (команда :set number редактора vim).</p> <p>Данный файл будет использоваться утилитой mdadm для автоматической активации RAID массивов во время загрузки системы и создания устройства /dev/md1.</p> <p>3. Проверить активность процесса создания RAID массивов, используя файл /proc/mdstat.</p>
5. Создать физические тома LVM на созданных RAID массивах.	<ol style="list-style-type: none"> 1. Создать физический том на основе устройства /dev/md0. 2. Создать физический том на основе устройства /dev/md1.
6. Создать группы томов LVM1 и LVM2 и просмотреть информацию о созданных группах.	<ol style="list-style-type: none"> 1. Создать группу LVM1. 2. Создать группу LVM2. 3. Просмотреть служебную информацию о созданных группах LVM. <p>Сделать соответствующий вывод: _____</p> <p>_____</p> <p>_____</p>
7. Создать логические тома в группах томов LVM1 и LVM2.	<ol style="list-style-type: none"> 1. Создать логический том ftp1 размером 50 Мбайт в группе томов LVM1. 2. Создать логический том ftp2 размером 50 Мбайт в группе томов LVM2.
8. Создать файловые системы ext3 на логических томах ftp1 и ftp2, смонтировать их и зафиксировать сделанные изменения в файле /etc/fstab.	<ol style="list-style-type: none"> 1. Создать файловые системы ext3 на логических томах ftp1 и ftp2. 2. Смонтировать созданные файловые системы в точки монтирования /pub1 и /pub2. 3. Зафиксировать сделанные изменения в файле /etc/fstab. 4. Выполнить перезагрузку системы и убедиться, что после перезагрузки созданные разделы /pub1 и /pub2 доступны.

Задания	Описание действий
<p>9. Расширить раздел /pub2 и файловую систему, содержащуюся на нем, до максимально возможного размера.</p>	<p>1. Расширить раздел /pub2 до максимально возможного размера _____ Мб. <pre># lvextend -l +11 /dev/LVM2/ftp2 -v</pre></p> <p>2. Расширить файловую систему, содержащуюся на логическом томе ftp2: <pre># resize2fs /dev/LVM2/ftp2</pre></p> <p>3. Просмотреть параметры группы логических томов LVM2 после расширения. Сделать соответствующий вывод: _____ _____ _____</p>
<p>10. Проверить отказоустойчивость массива RAID1. Для проверки целостности данных в разделе /pub2 необходимо создать файл /pub2/testfile, и определить его MD5 хэш. После того, как будет произведен тестовый отказ и восстановление, повторно определить MD5 хэш файла testfile и сравнить его с предыдущим полученным значением.</p>	<p>1. Создать файл testfile в разделе /pub2 размером 70Мбайт: <pre># dd if=/dev/zero of=/pub2/testfile bs=1k count=70000</pre></p> <p>2. Определить MD5 хэш файла /pub2/testfile и сохранить его в файле: <pre># md5sum /pub2/testfile >/tmp/testfile_1.md5</pre></p> <p>3. Выполнить пробный отказ устройства /dev/sde в массиве RAID1: <pre># mdadm /dev/md1 -f /dev/sde1</pre> <pre># cat /proc/mdstat</pre> <p>После того, как был произведен тестовый отказ устройства /dev/sde1, пользователь root должен получить соответствующее почтовое уведомление, а в файле /proc/mdstat отказавшее устройство должно быть помечено символом (F). Сделать соответствующий вывод: _____ _____ _____</p> <p>4. Убедиться в целостности файла /pub2/testfile можно, выполнив следующие команды: <pre># md5sum /pub2/testfile >/tmp/testfile_2.md5</pre> <pre># mp /tmp/testfile_1.md5 /tmp/testfile_2.md5</pre> <p>Вывод команды cmp не содержит данных, следовательно, хеши MD5 файла testfile до и после отказа идентичны.</p> </p></p>

Упражнение 1.6. Синхронизация данных

Сценарий: В вашей организации открылся дополнительный филиал «AU», находящийся в Австралии и имеющий медленный (~256Kbps) и не очень надежный канал связи с головным офисом. У руководства организации есть требование, чтобы данные всех пользователей удаленного филиала автоматически синхрони-

зировались на основную площадку в целях резервного копирования. Вам поставлена задача организовать защищенную синхронизацию данных между удаленным филиалом и основной площадкой, используя **rsync**. В качестве rsync-сервера, который будет использоваться для хранения резервных копий, используется виртуальная машина **srv.linux.lab**. В качестве клиента, с которого осуществляется резервирование данных, используется виртуальная машина **client.linux.lab**.

Задания	Описание действий
<p>1. Настроить сервис rsyncd, на сервере который будет использоваться для хранения резервируемых данных.</p>	<p>1. Зайти в систему под пользователем root.</p> <p>2. Убедиться, что в системе установлены пакеты stunnel и rsync, используя менеджер пакетов rpm. Установить данные пакеты в случае их отсутствия, используя менеджер пакетов yum.</p> <p>3. Создать конфигурационный файл /etc/rsyncd.conf следующего содержания:</p> <pre> syslog facility = local5 use chroot = yes uid = root gid = root pid file = /var/run/rsyncd.pid log file = /var/log/rsyncd.log max connections = 10 timeout = 600 read only = yes [au] path = /backup/au comment = AU office backup hosts allow = 127.0.0.1 read only = no list = yes ignore nonreadable = yes dont compress = * exclude = lost+found aquota.user .bash* </pre> <p>В данном случае резервирование данных филиала «AU» будет осуществляться в каталог /backup/au, который необходимо создать; в директиве exclude указаны образцы имен файлов, которые не будут копироваться на сервер rsync. В директиве log file указан файл, в который будет осуществляться запись системных событий демона rsyncd. Журналирование системных событий будет осуществляться с уровнем local5. Данную информацию необходимо добавить в файл /etc/syslog.conf и перезапустить сервис syslogd:</p> <pre> # echo "local5.* /var/log/rsyncd.log" >> /etc/syslog.conf && service syslog restart </pre> <p>4. Скопировать с виртуальной машины преподавателя (main.linux.lab) инициализационный скрипт демона rsyncd и добавить его, используя утилиту chkconfig, в автозапуск системы, затем запустить демон rsyncd:</p> <pre> # scp test@main.linux.lab:/pub/rsyncd /etc/init.d # chkconfig --add rsyncd </pre>

Задания	Описание действий
	<p>Убедиться, что демон <code>rsyncd</code> успешно запустился, просмотрев запись в соответствующем журнале syslog:</p> <pre>2009/01/29 20:47:08 [19591] rsyncd version 2.6.8</pre>
<p>2. Настроить сервис <code>stunnel</code>, на сервере который будет использоваться для хранения резервируемых данных.</p>	<ol style="list-style-type: none"> Добавить следующую информацию о порте, который будет использоваться для обмена защищенными данными в файл /etc/services: <pre>ssyncd 273/tcp # rsync over stunnel</pre> Добавить в файл /etc/hosts.allow информацию о том, с какого IP-адреса разрешено принимать соединения: <pre>ssyncd : 192.168.137.3</pre> <p>Строка hosts allow = 127.0.0.1 в файле /etc/rsyncd.conf означает, что обмен незашифрованным трафиком возможен только с локального IP-адреса 127.0.0.1.</p> Создать конфигурационный файл /etc/stunnel/stunnel.conf следующего содержания: <pre>cert = /etc/stunnel/server.pem client = no pid = /var/run/stunnel.pid [ssync] accept = 273 connect = 873</pre> Создать файл, содержащий сертификат и закрытый ключ, который будет использоваться для шифрования трафика, принимаемого демоном rsyncd: <pre># cd /etc/pki/tls/certs # make server.pem Country Name (2 letter code) [GB]:RU State or Province Name (full name) [Berkshire]:Moscow Locality Name (eg, city) [Newbury]:Moscow Organization Name (eg, company) [My Company Ltd]:SOFTLINE Organizational Unit Name (eg, section) []:<Enter> Common Name (eg, your name or your server's hostname) []:<srv.linux.lab> Email Address []:<Enter></pre> <p>В результате в текущем рабочем каталоге должен быть создан файл server.pem, содержащий закрытый ключ и сертификат.</p> Скопировать данный сертификат в каталог /etc/stunnel и установить на него код доступа 0400. Скопировать с виртуальной машины преподавателя (main.linux.lab) <code>init</code>-скрипт демона stunnel и добавить его, используя утилиту chkconfig, в автозапуск системы, затем запустить демон stunnel. Убедиться, что демон stunnel ожидает входящих соединений на заданном порте: <pre># netstat -tanpu grep 273 tcp 0 0 0.0.0.0:273 0.0.0.0:* LISTEN 21636/stunnel</pre>

Задания	Описание действий
<p>3. Настроить сервис stunnel на клиенте, с которого будет осуществляться резервирование данных.</p>	<ol style="list-style-type: none"> 1. Добавить следующую информацию о порте, который будет использоваться для обмена защищенными данными в файл /etc/services: <pre>ssync 273/tcp # rsync over stunnel</pre> 2. Добавить в файл /etc/hosts.allow информацию о том, с какого IP-адреса разрешено принимать соединения: <pre>ssync : LOCAL</pre> 3. Создать конфигурационный файл /etc/stunnel/stunnel.conf следующего содержания: <pre>cert = /etc/stunnel/client.pem client = yes pid = /var/run/stunnel.pid [ssync] accept = 873 connect = srv.linux.lab:273</pre> 4. Создать сертификат, который будет использоваться для шифрации трафика, принимаемого демоном rsyncd аналогичным способом, описанным в пункте 4 третьего задания. 5. Скопировать с виртуальной машины преподавателя (main.linux.lab) init-скрипт демона stunnel и добавить его, используя утилиту chkconfig, в автозапуск системы, затем запустить демон stunnel. 6. С помощью команды netstat убедиться, что демон stunnel ожидает входящих соединений на заданном порте.
<p>4. Выполнить синхронизацию домашних каталогов пользователей и убедиться в создании резервных копий.</p>	<ol style="list-style-type: none"> 1. На клиенте, используя команду rsync, осуществить синхронизацию данных. 2. На сервере, где функционирует демон rsyncd, убедиться, что резервные копии успешно создались. В журнальном файле /var/log/rsyncd.log должны присутствовать записи следующего содержания: <pre>2009/01/30 20:31:10 [28697] connect from main.linux.lab (127.0.0.1) 2009/01/30 20:31:10 [28697] rsync to au from main.linux.lab (127.0.0.1) 2009/01/30 17:31:10 [28697] sent 28 bytes received 297 bytes total size 153 2009/01/30 20:48:00 [29219] connect from main.linux.lab (127.0.0.1) 2009/01/30 20:48:01 [29219] rsync to au from main.linux.lab (127.0.0.1) 2009/01/30 17:48:01 [29219] home/</pre>
<p>5. Добавить команду синхронизации домашних каталогов в расписание cron.</p>	<ol style="list-style-type: none"> 1. Используя команду crontab -e, добавить запуск команды синхронизации файлов, которая использовалась в предыдущем задании, для ежедневного выполнения в 22 часа. <pre># crontab -e 0 22 * * * /usr/bin/rsync -a -vv -z /home localhost::au 2>\$1</pre>

Упражнение 1.7. Централизованное хранилище журнальных файлов

Сценарий: Вы работаете в крупном датацентре, в котором функционируют порядка 100 веб-серверов. С целью упрощения труда административного персонала и консолидации журнальных файлов в едином месте, вам поставлена задача организовать централизованное хранилище журнальных файлов, в которое будет осуществляться запись всех системных событий, происходящих на веб-серверах датацентра. В данной работе вы выполните тестирование сетевого журналирования системных событий, используя сервис **syslog**. В качестве хранилища журнальных файлов используется виртуальная машина **srv.linux.lab**. В качестве тестового клиента, с которого будет осуществляться посылка тестовых сообщений сервису **syslog**, используется виртуальная машина **client.linux.lab**.

Задания	Описание действий
1. Настроить сервис syslog на виртуальной машине srv.linux.lab .	1. Настроить сервис syslog на прием сетевых сообщений, добавив в конфигурационный файл /etc/sysconfig/syslog следующую запись: <pre>SYSLOGD_OPTIONS="-r"</pre> 2. Настроить сервис syslog на запись системных событий, с уровнем важности local5.* в файл /var/log/remote.log , добавив в конфигурационный файл /etc/syslog.conf следующую запись: <pre>local5.* /var/log/remote.log</pre> 3. Перезапустить сервис syslog и убедиться, что он ожидает входящих TCP соединений на 514 порту.
2. Настроить сервис syslog на виртуальной машине client.linux.lab .	1. Настроить сервис syslog на запись системных событий, с уровнем важности local5.* на виртуальную машину srv.linux.lab , добавив в конфигурационный файл /etc/syslog.conf следующую запись: <pre>local5.* @srv.linux.lab</pre> 2. Перезапустить сервис syslog .
3. Выполнить тестирование сетевого журналирования событий.	1. На виртуальной машине client.linux.lab , используя команду logger , послать на сервис syslog несколько тестовых сообщений с уровнем важности local5.debug . 2. На виртуальной машине srv.linux.lab , проверить, что тестовые события отразились в журнальном файле /var/log/remote.log .

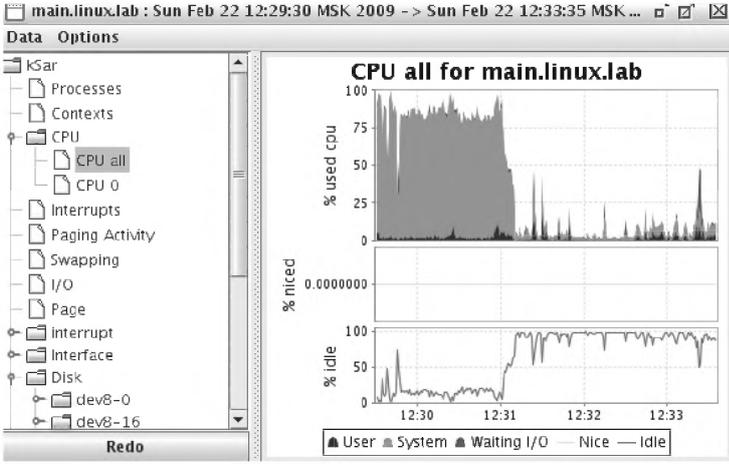
Упражнение 1.8. Мониторинг системных ресурсов

Сценарий: В целях повышения качества работы и доступности сетевых сервисов, реализованных на основе ОС Linux, а также предупреждения возможных сбоев в их работе, вы решили организовать регулярный сбор статистики о состоянии системных ресурсов при помощи утилиты `sar`. Каждый день вы хотите получать отчеты по собранным данным для того, чтобы их анализировать. Для удобства анализа вы хотите иметь графическое представление интересующих данных. В данной работе вам необходимо настроить сбор статистики по использованию системных ресурсов, таких как утилизация процессора, загрузка процессора, количество активных процессов, объем свободной виртуальной памяти, и время ожидания выполнения операций дискового ввода-вывода. Полученные данные необходимо проанализировать, используя утилиту `ksar`.

Предварительные требования

Для выполнения данного упражнения понадобится две виртуальных машины: **srv.linux.lab**, на которой будет настроен сбор статистики и **client.linux.lab**, которая будет выполнять роль полезной нагрузки.

Задания	Описания действий
<p>1. Настроить сбор статистических данных на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root в графическую оболочку GNOME.</p> <p>2. Установить пакет sysstat и jre-6u12-linux-i586.rpm. Пакет jre-6u12-linux-i586.rpm доступен по FTP с виртуальной машины преподавателя.</p> <p>3. Настроить выполнение команды сбора статистики каждую минуту, используя <code>crontab</code> файл <code>/etc/cron.d/sysstat</code>, и перезапустить демон <code>cron</code>. Убедиться, что данные собираются, используя команду sar.</p> <p>4. Установить утилиту kSar и запустить ее, используя файл run.sh. Дистрибутив данной утилиты располагается на машине преподавателя и доступен по FTP. Прежде чем запускать скрипт <code>run.sh</code>, определите корневой каталог, где располагаются файлы пакета jre-1.6.0_12-fcs, в переменной окружения JAVA_HOME.</p> <p>5. Сформировать файл urls, содержащий URL адреса документов из упражнения 2.4. На каждой строке данного файла должен располагаться один адрес URL, например:</p> <pre>http://srv.linux.lab/man/man.html http://srv.linux.lab/man/ulimits.gz http://srv.linux.lab/man/xinetd.gz</pre> <p>В данном примере представлено три URL адреса, каждый из которых располагается на отдельной строке. Для формирования данного файла, рекомендуется использовать команду lynx с ключом -dump.</p>

Задания	Описания действий
<p>2. Организовать нагрузочное тестирование и проанализировать использование системных ресурсов виртуальной машины srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Установить утилиту siege из исходного кода, дистрибутив которого располагается на машине преподавателя и доступен по FTP. Ознакомиться с синтаксисом запуска данной команды, используя руководство man по данной команде.</p> <p>3. На виртуальной машине srv.linux.lab в окне программы kSar зайти в меню Data -> Launch local command и указать команду sar -A 1 600.</p> <p>4. На виртуальной машине client.linux.lab, используя утилиту siege, запустить нагрузочное тестирование веб-сервера Apache, находящегося на виртуальной машине srv.linux.lab. В аргументах данной команды указать количество одновременных обращений равное 50 и файл urls, из которого будут формироваться веб-запросы.</p> <p>5. Спустя 5 минут, остановить процесс siege.</p> <p>6. Дождаться, пока в левой области окна программы kSar не отобразится статистическая информация. Примерный вид окна программы kSar представлен на следующем рисунке:</p>  <p>7. Проанализировать изменения следующих системных параметров:</p> <ul style="list-style-type: none"> • количество активных процессов _____; • количество прерываний _____; • средняя нагрузка на процессоре _____; • объем свободной памяти _____; • количество страниц виртуальной памяти считанных с диска и записанных на диск _____; • объем памяти использованной под дисковый кеш и буферы _____; • количество операций ввода-вывода, находящихся в очереди на исполнение _____;

Задания	Описания действий
	<ul style="list-style-type: none"> • простой процессора, связанный с операциями дискового ввода-вывода _____; • средняя частота пропуска сетевого интерфейса, через который осуществлялось нагрузочное тестирование _____.

Самостоятельные упражнения и дополнительные вопросы

1. Опишите более детально основные этапы загрузки ОС Linux на основе содержимого файла `/var/log/dmesg`.
2. Определите количество общей и резидентной виртуальной памяти, используемой процессами сервиса NFS, используя утилиту `top`.
3. Переместите файл `/etc/inittab` в произвольное место файловой системы, выполните перезапуск системы. Что произойдет? Перенесите файл `inittab` обратно в каталог `/etc`, используя первый уровень выполнения. При переносе файла `inittab` обратно в каталог `/etc` учтите, что корневой раздел на данном этапе доступен только в режиме чтения.
4. Просмотрите конфигурационный файл `kickstart`, сделанный в процессе инсталляции ОС Linux программой `anaconda` и определите какие группы пакетов были установлены в системе.
5. Опишите основные этапы создания логических томов **LVM** и **RAID** массивов.
6. Найдите три способа, которыми можно заблокировать учетную запись пользователя.
7. Выполните синхронизацию произвольного файла большого размера (~500Мбайт) при помощи команды `rsync` между виртуальными машинами `srv.linux.lab` и `client.linux.lab`. В процессе синхронизации отключите сетевой интерфейс у любой из виртуальных машин. Проанализируйте результат выполнения команды `rsync`. Запустите команду `rsync` повторно, так, чтобы, в случае повторного выключения сетевого интерфейса, синхронизируемый файл не был скопирован в место своего назначения.
8. Опишите общий принцип работы механизма виртуальной памяти ОС Linux.
9. Перечислите основные метрики оценки использования ресурсов процессора.
10. Какая из подсистем ОС Linux является наиболее подверженной к быстрому ухудшению производительности?
11. Каким образом можно изменить приоритет выполнения запущенного процесса?
12. Как определить объем виртуальной памяти, который использует процесс?

Практическая работа 2. Сетевое администрирование ОС Linux

Упражнение 2.1. Сетевая аутентификация пользователей LDAP

Сценарий: Компания, в которой вы работаете, имеет большое количество пользователей. В настоящее время все пользователи аутентифицируются на своих рабочих станциях, используя локальные файлы `/etc/passwd` и `/etc/shadow`. В целях консолидации учетных данных пользователей и сокращения административных затрат, вам поставлена задача организовать систему централизованной аутентификации пользователей с использованием каталога LDAP. Организация системы аутентификации пользователей производится в тестовой зоне, с использованием тестовой учетной записи пользователя **petrov**. Результатом выполнения данной работы является удачная аутентификация пользователя **petrov** в системе.

Задания	Описание действий
<p>1. Установить и настроить сервис каталогов LDAP на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под учетной записью пользователя root.</p> <p>2. Установить пакеты openldap, openldap-servers, openldap-clients и db4-utils, используя менеджер пакетов yum.</p> <p>3. Настроить основные параметры каталога LDAP в конфигурационном файле <code>/etc/openldap/slapd.conf</code>:</p> <pre> database bdb suffix "dc=example,dc=com" rootdn "cn=administrator,dc=example,dc=com" rootpw {MD5}abq1J7dhLu4BQFH19y7Alg== directory /var/lib/ldap </pre> <p>Пароль учетной записи administrator записан в виде хеша MD5, который необходимо получить при помощи команды slappasswd. В данном случае для хранения данных каталога используется база данных Berkeley (bdb), корневым каталогом в данной структуре каталогов является каталог example.com, в котором содержатся все остальные каталоги, включая каталог с учетными записями пользователей. Административной учетной записью данного каталога является пользователь administrator. С помощью данной записи возможно будет выполнять операции поиска и изменения данных в каталоге LDAP.</p> <p>Файлы базы данных LDAP каталога располагаются в каталоге <code>/var/lib/ldap</code>.</p> <p>4. Создать в каталоге <code>/var/lib/ldap</code> файл DB_CONFIG, содержащий системные параметры базы данных каталога LDAP.</p>

Задания	Описание действий
<p>2. Добавить тестовую учетную запись пользователя и группы в каталог LDAP и выполнить проверку их доступности.</p>	<p>1. Создать файл в формате LDIF (data.ldif), содержащий описание корневого каталога, организационных единиц (users и groups), в которых будут храниться учетные записи пользователей и групп. Пароли пользователя и группы необходимо предварительно создать, используя команду slappasswd. Для создания файла использовать образец data.ldif, доступный по FTP с виртуальной машины преподавателя.</p> <pre> dn: dc=example,dc=com dc: example objectClass: top objectClass: domain objectClass: domainRelatedObject associatedDomain: example.com dn: ou=users,dc=example,dc=com ou: users objectClass: top objectClass: organizationalUnit objectClass: domainRelatedObject associatedDomain: example.com dn: ou=group,dc=example,dc=com ou: group objectClass: top objectClass: organizationalUnit objectClass: domainRelatedObject associatedDomain: example.com dn: uid=petrov,ou=users,dc=example,dc=com uid: petrov cn: petrov sn: petrov mail: petrov@example.com objectClass: person objectClass: organizationalPerson objectClass: inetOrgPerson objectClass: posixAccount objectClass: top objectClass: shadowAccount userPassword: {MD5}I2VYp+wz4yI9tEcQJIMwEw== shadowLastChange: 14291 shadowMax: 99999 shadowWarning: 7 loginShell: /bin/bash uidNumber: 700 gidNumber: 701 homeDirectory: /home/petrov dn: cn=petrov,ou=group,dc=example,dc=com objectClass: posixGroup objectClass: top cn: petrov userPassword: {MD5}I2VYp+wz4yI9tEcQJIMwEw== gidNumber: 701 </pre>

Задания	Описание действий
	<p>В данном случае пользователь petrov будет иметь идентификатор равный 700, а соответствующая ему группа (petrov) будет иметь идентификатор равный 701. Следует также отметить, что пароли пользователя и группы заданы в формате MD5. Для задания пароля используется команда slappasswd -h {md5}.</p> <p>2. Экспортировать созданный LDIF файл (data.ldif) в каталог LDAP, используя команду slapadd. В выводе данной команды записи, начинающиеся со слова added:, свидетельствуют об удачном добавлении данных в каталог LDAP.</p> <p>3. Сделать системного пользователя ldap владельцем всех файлов базы данных каталога LDAP. Изменить атрибут группы для всех файлов базы данных каталога LDAP на группу ldap.</p> <p>4. Выполнить проверку конфигурационного файла slapd.conf, используя команду slaptest. В случае корректности всех сделанных ранее действий на терминале будет выведено сообщение, содержащее слово succeeded.</p> <p>5. Запустить демон slapd и настроить его автозапуск на 3 и 5 уровнях выполнения, используя команду chkconfig.</p>
<p>3. Настроить клиент аутентификации на виртуальной машине client.linux.lab и выполнить проверку аутентификации.</p>	<p>1. Установить на виртуальной машине client.linux.lab пакеты openldap-clients, openldap и nss-ldap.</p> <p>2. Отредактировать конфигурационный файл /etc/ldap.conf, добавив в него следующие строки:</p> <pre>host srv.linux.lab base dc=example,dc=com scope sub suffix "dc=example,dc=com" ssl no timelimit 120 bind_timelimit 120 idle_timelimit 3600 pam_filter objectclass=posixAccount pam_login_attribute uid pam_password md5 nss_base_passwd ou=users,dc=example,dc=com nss_base_shadow ou=users,dc=example,dc=com nss_base_group ou=group,dc=example,dc=com</pre> <p>3. Отредактировать конфигурационный файл /etc/openldap/ldap.conf, добавив в него следующие строки:</p> <pre>URI ldap://172.16.0.2 BASE dc=example,dc=com</pre>

Задания	Описание действий
	<p>4. Отредактировать конфигурационный файл /etc/nsswitch.conf, в котором указать сервис ldap как второстепенный источник получения информации по учетным данным.</p> <p>5. Выполнить проверку возможности получения информации об учетных данных из каталога LDAP, используя команду getent.</p> <p>6. Проверить доступность учетной записи тестового пользователя в созданном катлоге LDAP, используя утилиту ldapsearch.</p> <p>В результате выполнения команды ldapsearch должна отобразиться запись созданного пользователя.</p> <p>7. Проверить доступность учетной записи тестового пользователя в созданном катлоге LDAP, используя утилиты id и finger.</p> <p>8. Заменить конфигурационный файл /etc/pam.d/system-auth аналогичным файлом с машины преподавателя (доступ по FTP).</p> <p>9. Выполнить вход (в графический и текстовый режимы) в систему под учетной записью пользователя petrov и убедиться, что пользователь успешно аутентифицировался в системе, просмотрев журнальный файл /var/log/secure.</p>

Упражнение 2.2. Настройка монтирования домашнего каталога пользователя по требованию

Сценарий: Компания, в которой вы работаете, направляет вас в новый дополнительный офис с целью настройки пользовательских рабочих мест. Основным из требования является возможность каждого пользователя иметь постоянный доступ к своим личным данным, при этом, данные должны регулярно резервироваться. Для решения данной задачи вам предстоит настроить централизованное хранилище домашних каталогов пользователей, а так же их автоматическое монтирование при обращении пользователей к данным, находящихся в домашних каталогах. Результат выполнения данного упражнения является настроенный сервер NFS, предоставляющий доступ пользователей к своим домашним каталогам по их требованию.

Задания	Описания действий
<p>1. Настроить на виртуальной машине srv.linux.lab сервис NFS, предоставляющий доступ к домашним каталогам пользователей.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab от пользователя root и запустить сеанс командной строки.</p> <p>2. Создать каталог, в котором будут находиться персональные данные пользователя. В данном примере тестовый пользователь имеет регистрационное имя andrew, а каталог, в котором содержатся его данные называется /data/users/andrew.</p> <p>3. Добавить в файл /etc/exports файловую систему, к которой возможно будет получить доступ на чтение и запись данных по NFS:</p> <pre style="margin-left: 40px;">/data/users/andrew client.linux.lab(rw)</pre> <p>В данном случае для хоста client.linux.lab каталог /data/users/andrew будет доступен в режиме чтения/записи.</p> <p>4. Экспортировать файловую систему, указанную в файле exports, используя команду exportfs.</p> <p>5. Настроить запуск сервиса nfs на 3 и 5 уровнях выполнения, используя команду chkconfig, и запустить сервис nfs.</p>
<p>2. Настроить сервис autofs на виртуальной машине client.linux.lab для монтирования домашних каталогов пользователей по требованию.</p>	<p>1. Добавить в основной конфигурационный файл сервиса autofs строку, согласно которой будет выполняться монтирование файловых систем, указанных в файле /etc/auto.home в каталог /home.</p> <p>2. Добавить в файл /etc/auto.home команду монтирования каталога /data/users/andrew с хоста srv.linux.lab в точку монтирования andrew в режиме чтения/записи.</p> <p>3. Создать тестового пользователя andrew.</p> <p>4. Перезапустить сервис autofs, используя команду service.</p> <p>5. Настроить запуск сервиса autofs на 3 и 5 уровнях выполнения, используя команду chkconfig.</p>
<p>3. Выполнить проверку монтирования домашнего каталога тестового пользователя на виртуальной машине client.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной client.linux.lab машине под учетной записью пользователя andrew.</p> <p>2. Попробовать создать произвольный файл в текущем рабочем каталоге и убедиться что файл создан.</p> <p>3. От пользователя root выполнить команду /etc/auto.net srv.linux.lab для проверки доступных для монтирования файловых систем.</p> <p>Из вывода данной команды должно быть видно, какие каталоги доступны для монтирования на хосте srv.linux.lab.</p> <p>4. От пользователя andrew запустить команду mount и убедиться, что его домашний каталог подмонтирован по NFS.</p>

Упражнение 2.3. Настройка динамического обновления записей DNS

Сценарий: В локальной сети, администратором которой вы являетесь, присутствует 1000 клиентских хостов. Каждый хост имеет свое уникальное DNS имя и IP-адрес, получаемый от сервера DHCP. Согласно требованиям, на сервере DHCP максимальное время жизни отданных клиентам IP-адресов, составляет не более часа. Перед вами поставлена задача настроить динамическое добавление новых записей на DNS сервере по мере того, как будут распределяться IP-адреса между клиентскими хостами в локальной сети. В данной работе вам предстоит создать две тестовые DNS зоны - прямую и обратную, затем настроить на сервере DHCP пул IP-адресов, из которого клиентские хосты будут получать IP-адреса и, наконец, организовать динамическое обновление DNS записей клиентских хостов. Результатом выполнения данного упражнения является настроенный механизм динамического обновления DNS записей. В качестве серверов DNS и DHCP будет выступать виртуальная машина **srv.linux.lab**. В качестве DHCP клиента будет выступать виртуальная машина **client.linux.lab**.

Задания	Описания действий
<p>1. Настроить на виртуальной машине srv.linux.lab DNS сервер.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root и установить пакет bind.</p> <p>2. Создать конфигурационные файлы /etc/named/named.conf и /etc/named/named.root.hints, используя образцы данных файлов, находящихся в каталоге /usr/share/doc/bind-<версия>/sample/etc.</p> <p>3. Добавить в файл named.conf описания прямой и обратной зоны (example.com и 0.16.172.in-addr.arpa) в представление internal:</p> <pre>view "internal" { zone "linux.lab" { type master; allow-update { ANY; }; file "data/linux.lab.db"; notify yes; }; zone "0.16.172.in-addr.arpa" { type master; allow-update { ANY; }; file "data/0.16.172.in-addr.arpa.db"; notify yes; }; };</pre> <p>4. Изменить в представлении internal значения директив match-clients и match-destinations, а также изменить значение директивы include для корневой зоны:</p>

Задания	Описания действий
	<pre> match-clients { ANY; }; match-destinations { ANY; }; include "/etc/named/named.root.hints"; </pre> <p>5. Запустить команду rndc-confgen. Из части вывода команды rndc-confgen (от строки # Start of rndc.conf до строки # End of rndc.conf) создать файл /etc/named/rndc.conf. Часть вывода команды rndc-confgen (от строки # Use with the following до строки # End of named.conf) добавить в файл /etc/named/named.conf перед директивой view.</p> <p>6. Закомментировать все остальные представления в файле named.conf.</p> <p>7. Проверить файл named.conf на отсутствие ошибок, используя команду named-checkconf. В случае отсутствия ошибок код завершения данной команды будет равен нулю.</p> <p>8. Добавить в файле /etc/sysconfig/named в переменную OPTIONS опцию -c, указывающую на расположение файла named.conf.</p> <p>9. Создать файлы прямой и обратной зоны в каталоге /var/named/data, используя за образец файл /usr/share/doc/bind-<версия>/sample/var/named/localhost.zone.</p> <p>10. Создать файл корневой зоны named.root в каталоге /var/named, используя за образец файл /usr/share/doc/bind-<версия>/sample/var/named/named.root.</p> <p>11. Проверить файлы прямой и обратной зоны на отсутствие ошибок, используя команду named-checkzone:</p> <pre> # named-checkzone linux.lab /var/named/data/linux.lab.db zone linux.lab/IN: loaded serial 42 OK # named-checkzone 0.16.172.in-addr.arpa /var/named/data/0.16.172.in-addr.arpa.db zone 0.16.172.in-addr.arpa/IN: loaded serial 42 OK </pre> <p>12. Назначить пользователя named и группу named владельцем и группой всех файлов, находящихся в каталоге /var/named.</p> <p>13. Скопировать директиву key из файла named.conf в файл /etc/named/rndc.key.</p> <p>14. Запустить демон named и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig. В случае корректного запуска демона named, в файле /var/log/messages должны быть выведены сообщения о загрузке файлов прямой</p>

Задания	Описания действий
	<p>и обратной зон в память:</p> <pre>named[11388]: zone 0.16.172.in-addr.arpa/IN/ internal: loaded serial 42 main named[11388]: zone linux.lab/IN/internal: loaded serial 42 named[11388]: running</pre>
<p>2. Выполнить проверку обновления зон DNS на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить проверку обновления зон, используя утилиты nsupdate и dig:</p> <pre># nsupdate > update add client.linux.lab 300 A 172.16.0.50 > <Enter> > quit [root@main named]# dig @172.16.0.2 client.linux. lab A ; <<>> DiG 9.3.4-P1 <<>> @172.16.0.2 client.linux. lab A ; (1 server found) ;; global options: printcmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: ;; ANSWER SECTION: client.linux.lab. 300 IN A 172.16.0.50</pre>
<p>3. Настроить на виртуальной машине srv.linux.lab сервис DHCP.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root и установить пакет dhcp.</p> <p>2. Создать конфигурационный файл /etc/dhcpd/dhcpd.conf, используя его образец в каталоге /usr/share/doc/dhcp-<версия>/dhcpd.conf.sample. Итоговый вид файла dhcpd.conf должен иметь следующий вид:</p> <pre>authoritative; ddns-update-style interim; ddns-updates on; allow client-updates; include "/etc/named/rndc.key"; zone linux.lab. { primary 127.0.0.1; key "rndckey"; } subnet 172.16.0.0 netmask 255.255.255.0 { option routers 172.16.0.2; option subnet-mask 255.255.255.0; option domain-name-servers 172.16.0.2; option domain-name "linux.lab."; ddns-domainname "linux.lab."; ddns-rev-domainname "in-addr.arpa."; option broadcast-address 172.16.0.255; range 172.16.0.10 172.16.0.100;</pre>

Задания	Описания действий
	<pre> default-lease-time 86400; max-lease-time 172800; host ns { next-server srv.linux.lab; hardware Ethernet 00:0C:29:8E:69:6A; fixed-address 172.16.0.2; } } zone linux.lab. { primary 172.16.0.2; } zone 0.16.172.in-addr.arpa. { primary 172.16.0.2; } </pre> <p>3. Указать в файле /etc/sysconfig/dhcpd в переменной DHCPDARGS параметры используемого конфигурационного файла и сетевого интерфейса. Сетевой интерфейс должен находиться в локальной подсети 172.16.0.0/24.</p> <p>4. Запустить демон dhcpd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig. В случае корректного запуска демона dhcpd, в файле /var/log/messages должно присутствовать сообщение о приеме пакетов DHCP на указанном сетевом интерфейсе:</p> <pre>Feb 17 18:41:49 main dhcpd: Listening on LPF/eth0/00:0c:29:8e:69:6a/172.16.0/24</pre>
<p>4. Настроить клиента DHCP на виртуальной машине client.linux.lab и проверить динамическое обновление соответствующих DNS записей.</p>	<p>1. Выполнить вход в систему под пользователем root на виртуальной машине client.linux.lab.</p> <p>2. Настроить сетевой интерфейс на получение динамического IP-адреса, используя файл /etc/sysconfig/network-scripts/ifcfg-eth<N>. Параметр <N> соответствует номеру интерфейса:</p> <pre> DEVICE=eth1 ONBOOT=yes BOOTPROTO=dhcp </pre> <p>3. Настроить клиента DHCP, используя за образец конфигурационный файл /usr/share/doc/dhclient-<версия>/dhclient.conf. Итоговый конфигурационный файл /etc/dhclient.conf должен иметь следующий вид:</p> <pre> send host-name "<локальное_имя_хоста>"; request subnet-mask, broadcast-address, routers, domain-name, domain-name-servers, host- name; require subnet-mask, domain-name-servers; </pre>

Задания	Описания действий
	<pre> timeout 60; retry 60; reboot 10; select-timeout 5; initial-interval 2; script "/sbin/dhclient-script"; </pre> <p>Здесь в директиве send host-name необходимо указать локальное имя хоста (без доменной части), которое будет посылать DHCP-клиент DHCP-серверу. В данном случае в качестве имени хоста используется имя clientN.</p> <p>4. Получить IP-адрес от DHCP-сервера:</p> <pre> # dhclient -r eth1 # dhclient eth1 # service network restart </pre> <p>5. Проверить, что на DNS сервере создались A и PTR записи для данного клиента DHCP, используя команду dig.</p> <p>6. Убедиться, используя журнальный файл /var/log/messages на виртуальной машине srv.linux.lab, в том, что обновление DNS записей действительно произошло. В случае удачного обновления в данном файле должны присутствовать следующие строки:</p> <pre> dhcpd: Added new forward map from client.linux.lab. to 172.16.0.100 named[6146]: client 172.16.0.2#32771: view internal: updating zone '0.16.172.in-addr.arpa/IN': deleting rrset at '100.0.16.172.in-addr.arpa' PTR named[6146]: client 172.16.0.2#32771: view internal: updating zone '0.16.172.in-addr.arpa/IN': adding an RR at '100.0.16.172.in-addr.arpa' PTR named[6146]: zone 0.16.172.in-addr.arpa/IN/internal: sending notifies (serial 70) dhcpd: added reverse map from 100.0.16.172.in-addr. arpa. to client.linux.lab. dhcpd: DHCPREQUEST for 172.16.0.100 (172.16.0.2) from 00:0c:29:07:fc:33 (rle5) via eth0 dhcpd: DHCPACK on 172.16.0.100 to 00:0c:29:07:fc:33 (rhel5) via eth0 </pre>

Упражнение 2.4. Организация централизованного веб-доступа к документации ОС Linux.

Сценарий: Вы администрируете локальную сеть, в которой имеется большое количество серверов с ОС Linux. Время от времени вам приходится пользоваться

руководствами (HowTo) по ОС Linux, доступными на внешнем веб-ресурсе Linux Documentation Project. В целях сокращения времени на поиск необходимой информации вы решили организовать внутри локальной сети ресурс, который будет предоставлять веб-доступ к руководствам ОС Linux, включая **man** страницы основных команд. В целях снижения нагрузки на данный ресурс и повышения времени отклика вы решили установить дополнительно реверсный прокси сервер, который будет кешировать большинство повторяющихся веб-запросов к файлам документации. Для реализации поставленной цели вам понадобится настроить веб-сервер **Apache** и прокси сервер **Squid**.

Результатом выполнения данной работы будет являться веб-ресурс, содержащий руководства ОС Linux. Доступ к данному ресурсу должен осуществляться через веб-браузер. В качестве ресурса с документацией используется виртуальная машина **srv.linux.lab**. Проверка результата выполнения работы будет выполняться с клиента, которым является виртуальная машина **client.linux.lab**, а также непосредственно с виртуальной машины **srv.linux.lab**.

Задания	Описания действий
<p>1. Установить и настроить на виртуальной машине srv.linux.lab веб-сервер Apache и настроить доступ к документации ОС Linux, включая man страницы.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root и установить пакет httpd.</p> <p>2. Отредактировать конфигурационный файл httpd.conf следующим образом:</p> <ul style="list-style-type: none"> • Указать IP-адрес, на котором сервер будет принимать веб-запросы: <pre>Listen 127.0.0.1:80</pre> • Создать именованный виртуальный хост, на который будут непосредственно обращаться клиенты: <pre>NameVirtualHost 127.0.0.1:80 <VirtualHost 127.0.0.1:80> ServerAdmin root@localhost DocumentRoot "/var/www/html/" ServerName srv.linux.lab ServerAlias docs.linux.lab Alias /linux-docs /opt/docs ErrorLog logs/linux-docs-error_log CustomLog logs/linux-docs-access_log common <Directory /opt/docs> Options +Indexes AllowOverride None Order allow,deny Allow from all </Directory> </VirtualHost></pre> <p>3. Разместить каталоги с документацией и man-страницами в каталоге файловой системы /opt/docs. Для этого необходимо скопировать по FTP архивные файлы Linux-html-HOWTOs.tar.bz и man-html-*.tar.bz2</p>

Задания	Описания действий
	<p>с машины преподавателя и распаковать их в каталог /opt/docs.</p> <p>4. Выполнить проверку конфигурационного файла httpd.conf, используя команду httpd. В случае отсутствия ошибок, на терминал будет выведена конфигурация виртуального хоста и фраза Syntax OK.</p> <p>5. Запустить демон httpd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p> <p>6. Проверить возможность веб-доступа к содержимому каталога /opt/docs, зайдя по ссылке http://127.0.0.1/linux-docs на виртуальной машине srv.linux.lab. Проверку выполнять при помощи консольного веб-обозревателя lynx.</p> <p>7. В случае невозможности получить доступ к ресурсу http://127.0.0.1/linux-docs, проверить журнальные файлы linux-docs-error_log и linux-docs-error_log, находящиеся в каталоге /var/log/httpd и устранить ошибки самостоятельно.</p>
<p>2. Установить и настроить на виртуальной машине srv.linux.lab прокси сервер Squid и настроить его на кеширование входящих запросов.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root и установить пакет squid.</p> <p>2. Отредактировать конфигурационный файл squid.conf следующим образом:</p> <ul style="list-style-type: none"> • Указать IP-адрес, на котором сервер будет принимать входящие веб-запросы: <pre>http_port <IP-адрес>:80 vhost</pre> <p>Здесь в качестве параметра <IP-адрес> необходимо указать публичный IP-адрес виртуальной машины srvN.linux.lab (Интерфейс host-only) и порт прокси-сервера.</p> • Разрешить доступ для всех адресов из списка all и закомментировать все остальные правила доступа http_access: <pre>http_access allow all</pre> • Указать целевой веб-сервер, на который следует посылать веб-запросы при первом обращении: <pre>cache_peer 127.0.0.1 parent 80 0 originserver default</pre> • Установить значение директивы visible_hostname: <pre>visible_hostname srv.linux.lab</pre> <p>Данная директива будет уникально идентифицировать данный прокси сервер.</p> <p>3. Выполнить проверку конфигурационного файла squid.conf, используя команду squid. В случае</p>

Задания	Описания действий
	<p>отсутствия ошибок в файле squid.conf вывод данной команды должен быть пустым.</p> <p>4. Запустить демон squid и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p>
<p>3. Проверить механизм кеширования запросов.</p>	<p>1. На виртуальной машине srv.linux.lab выполнить проверку кеширования запросов, используя команду squidclient. Вместо параметра <IP-адрес> необходимо указать публичный IP-адрес виртуальной машины srv.linux.lab (Интерфейс host-only):</p> <ul style="list-style-type: none"> • Запустить команду squidclient и проанализировать заголовки X-Cache(_____) и X-Cache-Lookup(_____). • Повторно запустить команду squidclient и проанализировать заголовки X-Cache(_____) и X-Cache-Lookup(_____). <p>2. На виртуальной машине client.linux.lab, используя веб-браузер Mozilla Firefox, убедиться, что ресурс http://srv.linux.lab/linux-docs доступен и содержит руководства (HOWTO) ОС Linux, а также страницы руководства man.</p>

Упражнение 2.5. Настройка аутентификации SSH по публичному ключу

Сценарий: Вы администрируете локальную сеть, в которой имеется большое количество серверов с ОС Linux. Администрирование данных серверов осуществляется с использованием защищенного интерпретатора команд SSH. Каждый раз при подключении к серверу вам необходимо вручную вводить учетные данные пользователя и аутентифицироваться в системе. Вы не хотите держать в голове все те учетные данные, которые необходимы для подключения к серверам, и поэтому решили автоматизировать процесс аутентификации SSH путем настройки аутентификации с использованием открытых ключей. В данной работе вам необходимо настроить и протестировать механизм аутентификации SSH с использованием открытых ключей. Тестирование выполняется между клиентом, в качестве которого выступает виртуальная машина **client.linux.lab** и сервером, в качестве которого выступает виртуальная машина **serverN.linux.lab**.

Задания	Описания действий
<p>1. Настроить клиента ssh на виртуальной машине client.linux.lab.</p>	<ol style="list-style-type: none"> 1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root. 2. Сгенерировать открытый ключ (id_rsa.pub), используя команду ssh-keygen и настроить соответствующие права на каталог .ssh и его содержимое, а именно: <ul style="list-style-type: none"> • Удалить права на запись в каталоге ~/ для группы и всех остальных пользователей на каталог • Установить полные права на каталог ~/ssh только для его владельца. • Удалить все права для группы и всех остальных пользователей со всех файлов, находящихся в каталоге ~/ssh. 3. Скопировать открытый ключ (id_rsa.pub) на виртуальную машину srv.linux.lab, используя команду scp, и добавить его содержимое в файл ~/ssh/authorized_keys. 4. Настроить демон ssh-agent, добавив в файл ~/bash_profile запуск команды ssh-agent. Данный демон будет использоваться для кеширования запросов секретной фразы (passphrase) для последующих подключений по SSH. 5. Добавить закрытый ключ в кеш демона ssh-agent, используя команду ssh-add: <pre># ssh-add ~/.ssh/id_rsa Enter passphrase for /root/.ssh/id_rsa: <P@ssw0rd></pre>
<p>2. Настроить на виртуальной машине srv.linux.lab демона sshd на использование аутентификации по открытым ключам.</p>	<ol style="list-style-type: none"> 1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root. 2. Установить код доступа 600 на файл ~/ssh/authorized_keys. 3. Отредактировать конфигурационный файл /etc/ssh/sshd_config следующим образом: <ul style="list-style-type: none"> • Активизировать возможность аутентификации по открытым ключам: <pre>RSAAuthentication yes PubkeyAuthentication yes AuthorizedKeysFile .ssh/authorized_keys</pre> <p>В качестве параметра <IP-адрес> необходимо указать локальный IP-адрес виртуальной машины srv.linux.lab (Интерфейс host-only).</p> • Разрешить доступ по SSH для пользователя root: <pre>PermitRootLogin yes</pre> 4. Запустить демон sshd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.

Задания	Описания действий
	5. Запустить демон sshd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig .
3. Проверить работу механизма аутентификации SSH с использованием открытых ключей.	<p>1. На виртуальной машине client.linux.lab выполнить тестовое подключение по ssh к виртуальной машине srv.linux.lab:</p> <pre># ssh -v -o PreferredAuthentications=publickey srv.linux.lab debug1: Next authentication method: publickey debug1: Offering public key: /root/.ssh/id_rsa debug1: Server accepts key: pkalg ssh-rsa blen 277 debug1: Authentication succeeded (publickey).</pre> <p>В случае удачной аутентификации на терминале в отладочной информации будет присутствовать строка Authentication succeeded (publickey).</p> <p>2. На виртуальной машине srv.linux.lab проверить журнальный файл /var/log/secure на наличие записей, свидетельствующих о подключении клиента с использованием открытого ключа.</p>

Упражнение 2.6. Создание защищенного почтового сервиса на основе MTA Sendmail и Dovecot

Сценарий: В компании, в которой вы являетесь системным администратором, открылся небольшой дополнительный офис. Пользователи, находящиеся в дополнительном офисе, должны иметь защищенный доступ к своим почтовым ящикам по протоколу IMAPS, находящимися в главном офисе. Кроме того, пользователи должны иметь возможность отправлять почтовые сообщения через защищенное SMTP соединение, поскольку канал, соединяющий главный и дополнительный офис, проходит через сеть Интернет. В данном упражнении вам предстоит реализовать данный почтовый сервис на основе MTA Sendmail и ПО Dovecot, а затем протестировать созданную почтовую конфигурацию.

Задания	Описания действий
<p>1. Настроить MTA Sendmail на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Установить пакет sendmail-cf.</p> <p>3. Сгенерировать сертификат, который будет использоваться для шифрования SMTP соединения. В процессе генерации сертификата указать отмеченные жирным шрифтом значения:</p> <pre># cd /etc/pki/tls/certs # make sendmail.pem ... Country Name (2 letter code) [GB]:RU State or Province Name (full name) [Berkshire]: Moscow Locality Name (eg, city) [Newbury]:Moscow Organization Name (eg, company) [My Company Ltd]: SOFTLINE Organizational Unit Name (eg, section) []:IT Common Name (eg, your name or your server's hostname) []:srv.linux.lab E-mail Address []:root@linux.lab</pre> <p>4. Настроить конфигурационный файл sendmail.mc следующим образом:</p> <ul style="list-style-type: none"> • Сохранить исходную конфигурацию данного файла • Настроить прием входящих SMTP соединений на публичном IP-адресе: <pre>dn1 DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dn1</pre> • Запретить подключения пользователей, не использующих SSL: <pre>define(`confAUTH_OPTIONS', `A p')dn1</pre> • Определить действующие аутентификационные механизмы: <pre>TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dn1 define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dn1</pre> • Определить сертификат, который будет использоваться для шифрования: <pre>define(`CERT_DIR', `/etc/pki/tls/certs')dn1 define(`confCACERT_PATH', `CERT_DIR')dn1 define(`confCACERT', `CERT_DIR/sendmail.pem')dn1 define(`confSERVER_CERT', `CERT_DIR/sendmail. pem')dn1 define(`confSERVER_KEY', `CERT_DIR/sendmail. pem')dn1 define(`confCLIENT_CERT', `CERT_DIR/sendmail. pem')dn1 define(`confCLIENT_KEY', `CERT_DIR/sendmail. pem')dn1</pre>

Задания	Описания действий
	<ul style="list-style-type: none"> • Убедиться, что в данном файле присутствует опция FEATURE(use_cw_file)dnl. Добавить домен linux.lab в файл /etc/mail/local-host-names. 5. Добавить в файл /etc/mail/access домен linux.lab и разрешить данному домену отправлять почту. Затем сформировать хешированный файл данных access.db. 6. Создать двух тестовых пользователей andrew и kirll и задать для них пароль P@ssw0rd. 7. Сгенерировать конфигурационный файл sendmail.cf и перезапустить демон sendmail. 8. Проверить, что MTA Sendmail ожидает защищенных соединений на 25 порте: <pre> # openssl s_client -starttls smtp -crlf -connect 172.16.0.2:25 CONNECTED(00000003) --- 220 srv01.linux.lab ESMTP Sendmail 8.13.8/8.13.8; Fri, 20 Feb 2009 12:56:34 +0300 </pre> В случае удачной настройки MTA Sendmail, будет выдана команда с кодом 220. 9. Запустить демон sendmail и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.
<p>2. Настроить на виртуальной машине srv.linux.lab демон dovecot, который будет использоваться для защищенного доступа к почтовым ящикам.</p>	<ol style="list-style-type: none"> 1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root. 2. Установить пакет dovecot. 3. Запустить демон saslauthd, который будет использоваться в процессе аутентификации, и настроить его автозапуск, используя команды service и chkconfig. 4. Сохранить сертификат, входящий в состав пакета dovecot: <pre> # find /etc -name dovecot.pem -execdir mv {} dovecot.pem.orig \; </pre> 5. Настроить шаблон /etc/pki/dovecot/dovecot-openssl.cnf, который будет использован для создания нового сертификата: <pre> [req] default_bits = 1024 encrypt_key = yes distinguished_name = req_dn x509_extensions = cert_type prompt = no [req_dn] # country (2 letter code) C=RU # State or Province Name (full name) ST=Moscow </pre>

Задания	Описания действий
	<pre> # Locality Name (eg. city) L=Moscow # Organization (eg. company) O=SOFTLINE # Organizational Unit Name (eg. section) OU=IMAP server # Common Name (*.example.com is also possible) CN=srv.linux.lab # E-mail contact emailAddress=root@linux.lab [cert_type] nsCertType = server </pre> <p>Здесь необходимо указать значения, аналогичные пункту 3 первого задания, параметр OU оставить без изменений.</p> <p>6. Создать сертификат, который будет использоваться для защищенных подключений IMAP:</p> <pre> # /usr/share/doc/dovecot-1.0.7/examples/mkcert.sh a001 logout * BYE Logging out </pre> <p>7. Отредактировать конфигурационный файл /etc/dovecot.conf следующим образом:</p> <ul style="list-style-type: none"> • Сохранить исходную конфигурацию данного файла. • Добавить описание используемых протоколов и прослушиваемый IP-адрес: <pre> protocols = imaps listen = 172.16.0.2 </pre> • Указать параметры используемого сертификата: <pre> ssl_disable = no ssl_cert_file = /etc/pki/dovecot/certs/dovecot.pem ssl_key_file = /etc/pki/dovecot/private/dovecot.pem </pre> • Указать месторасположение почтовых ящиков пользователей: <pre> mail_location = mbox:~/mail:INBOX=/var/mail/%u </pre> • Указать параметры журналирования: <pre> log_path = /var/log/dovecot.log info_log_path = /var/log/dovecot-info.log </pre> • Добавить исправление известных ошибок почтовых клиентов: <pre> imap_client_workarounds = outlook-idle </pre> <p>8. Запустить демон dovecot и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p> <p>9. Убедится, что на указанном IP-адресе демон dovecot ожидает входящих соединений:</p> <pre> # openssl s_client -connect 172.16.0.2:993 * OK Dovecot ready. </pre>

Задания	Описания действий
<p>3. Настроить на виртуальной машине client.linux.lab почтовый клиент Mozilla Thunderbird.</p>	<p>1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root.</p> <p>2. Установить почтовый клиент Mozilla Thunderbird и настроить в нем два профиля IMAP для двух пользователей: andrew и kirill. Настройка осуществляется с учетом использования SSL при отправке и получении почты.</p> <p>3. Отправить несколько писем между данными пользователями и убедиться, что почтовые сообщения доставляются адресатам.</p>

Упражнение 2.7. Организация файлообменного сервиса FTP

Сценарий: Вы занимаетесь виртуальным веб-хостингом. Сервер, на котором находятся ресурсы ваших клиентов, располагается на удаленной площадке. Для обновления веб контента, расположенного на вашем сервере, клиенты подключаются к серверу, используя стандартный протокол FTP. Естественно, что при этом все учетные данные и файлы передаются в открытом виде, что крайне не безопасно. Кроме того, у каждого клиента должен быть свой логин и пароль, который предоставляет доступ на веб-сервер в соответствующий корневой каталог. Исходя из этого, вы решили организовать защищенную передачу данных FTP между вашим сервером и клиентами, а также настроить несколько виртуальных пользователей, которые будут использоваться вашими клиентами для управления веб-контентом. В данном упражнении вам предстоит реализовать задуманное на основе ПО VS-FTPD. В качестве FTP сервера будет выступать виртуальная машина **srv.linux.lab**, а клиентом будет являться виртуальная машина **client.linux.lab**.

Задания	Описания действий
<p>1. Настроить FTP-сервер на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Установить пакеты vsftpd и compat-db.</p> <p>3. Сгенерировать сертификат (vsftpd.pem), который будет использоваться для шифрования FTP соединения. Генерация сертификата выполняется аналогичным образом, как было показано в упражнении 2.6.</p> <p>4. Настроить модуль аутентификации /etc/pam.d/vsftpd на использование файла vsftpd_users для проверки учетных данных пользователей:</p>

Задания	Описания действий
	<pre> session optional pam_keyinit.so force revoke auth required /lib/security/pam_userdb.so db=/etc/vsftpd/vsftpd_users account required /lib/security/pam_userdb.so db=/etc/vsftpd/vsftpd_users </pre> <p>5. Создать виртуального пользователя virtualftp, в домашнем каталоге которого (/var/www/html/virtual) будут находиться каталоги клиентов. В качестве командного интерпретатора для данного пользователя задать значение /sbin/nologin.</p> <p>6. Создать текстовый файл (/etc/vsftpd/vsftpd_users.plain), содержащий учетные записи виртуальных пользователей:</p> <pre> customer1 P@ssw0rd customer2 qlqlql </pre> <p>В данном файле создано две учетные записи, логин и пароль каждой записи должен находиться на двух соседних строках.</p> <p>7. Создать хешированный файл данных, содержащий учетные записи виртуальных пользователей:</p> <pre> # db42_load -T -t hash -f /etc/vsftpd/vsftpd_users. plain /etc/vsftpd/vsftpd_users.db # chmod 600 /etc/vsftpd/vsftpd_users.db /etc/ vsftpd/vsftpd_users.plain </pre> <p>8. Создать необходимые каталоги для клиентов. Имена данных каталогов должны совпадать с регистрационными именами виртуальных пользователей.</p> <p>9. Настроить доступ на запись данных для виртуальных пользователей, с тем, чтобы они могли загружать веб-контент в свои каталоги:</p> <pre> # chown -R virtualftp:virtualftp /var/www/html/ virtual/ # chmod -R 644 /var/www/html/virtual/ # find /var/www/html/virtual -type d -exec chmod 755 {} \; </pre> <p>10. Настроить конфигурационный файл /etc/vsftpd/vsftpd.conf следующим образом:</p> <ul style="list-style-type: none"> • Разрешить доступ анонимным и локальным пользователям: <pre> anonymous_enable=YES local_enable=YES </pre>

Задания	Описания действий
	<ul style="list-style-type: none"> • Настроить корневой каталог для виртуальных пользователей: <pre>chroot_local_user=YES user_sub_token=\$USER local_root=/var/www/html/virtual/\$USER</pre> • Настроить права доступа и список доступа для локальных пользователей: <pre>write_enable=YES local_umask=022 userlist_enable=YES</pre> • Настроить учетную запись, которая будет использоваться виртуальными пользователями для доступа к своим каталогам: <pre>guest_enable=YES guest_username=virtualftp</pre> • Настроить права доступа для анонимных и виртуальных пользователей: <pre>anon_umask=0022 anon_upload_enable=YES anon_mkdir_write_enable=YES anon_other_write_enable=YES</pre> • Настроить системные параметры FTP, а также параметры SSL подключений: <pre>pam_service_name=vsftpd listen=YES ssl_enable=YES allow_anon_ssl=YES force_local_data_ssl=NO force_local_logins_ssl=NO ssl_tlsv1=YES ssl_sslv2=NO ssl_sslv3=NO rsa_cert_file=/etc/pki/tls/certs/vsftpd.pem</pre> <p>11. Убедиться, что демон vsftpd ожидает защищенных соединений на 21 порте.</p> <p>12. Запустить демон vsftpd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p>
<p>2. Проверить подключение FTP клиента, загрузку файлов на сервер и шифрацию данных FTP соединения, используя виртуальную машину client.linux.lab.</p>	<ol style="list-style-type: none"> 1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root. 2. Установить пакет wiresahrk-gnome. 3. В графической оболочке GNOME запустить анализатор пакетов Wireshark (Application -> Internet -> Wireshark Network Analyzer) на прослушивание сетевого интерфейса eth1. 4. Настроить клиента lftp на использование SSL подключений:

Задания	Описания действий
	<pre>set ftp:ssl-force yes set ssl:verify-certificate no set ftp:ssl-protect-data yes</pre> <p>5. Подключиться, используя клиент lftp к серверу srv.linux.lab и загрузить произвольный файл на FTP сервер.</p> <p>6. Проанализировать информацию, используя анализатор пакетов Wireshark, и убедиться, что FTP соединение шифруется и учетные данные пользователей не передаются в открытом виде.</p>

Упражнение 2.8. Синхронизации времени

Сценарий: В локальной сети, администратором которой вы являетесь, присутствует система мониторинга системных событий, связанных с производительностью, и событий, связанных с информационной безопасностью. Данная система формирует отчеты на основании собранных статистических данных. На основании этих отчетов вы периодически оцениваете производительность различных сетевых сервисов, компоненты которых физически развернуты на нескольких серверах. Для того чтобы сделать временную оценку загруженности какого-либо сервиса, вам необходимо, чтобы рассогласование времени на серверах, задействованных в данных сервисах, было минимальным. Для этого вы решили организовать локальный сервер времени NTP, с которым будут синхронизироваться все остальные сервера вашей локальной сети. В качестве локального NTP сервера будет выступать виртуальная машина **srv.linux.lab**, а сервером, который будет выполнять синхронизацию времени, будет являться виртуальная машина **client.linux.lab**.

Задания	Описания действий
<p>1. Настроить NTP сервер на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Установить пакет ntp.</p> <p>3. Настроить конфигурационный файл /etc/ntp.conf следующим образом:</p> <ul style="list-style-type: none"> • Запретить подключение по протоколу NTP для всех явно не указанных хостов: <pre>restrict default ignore</pre> • Указать хосты, которым будет разрешено синхронизировать время: <pre>restrict 172.16.0.0 mask 255.255.255.0 nomodify notrap nopeer restrict 127.0.0.1 restrict -6 ::1</pre>

Задания	Описания действий
	<ul style="list-style-type: none"> • Определить сервер, с которым будет выполняться синхронизация локального времени: <pre>server 127.127.1.0 fudge 127.127.1.0 stratum 10</pre> • Настроить системные параметры демона ntpd: <pre>driftfile /var/lib/ntp/drift logfile /var/log/ntp.log</pre> <p>4. Запустить демон ntpd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p> <p>5. Проверить синхронизацию времени с локальными часами, используя команду ntpq -p.</p>
<p>2. Настроить клиента NTP на виртуальной машине client.linux.lab и проверить синхронизацию времени.</p>	<p>1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root.</p> <p>2. Установить пакет ntp.</p> <p>3. Настроить конфигурационный файл /etc/ntp.conf следующим образом:</p> <ul style="list-style-type: none"> • Указать сервер, с которым будет выполняться синхронизация времени: <pre>server srv.linux.lab</pre> • Определить сервер, с которым будет выполняться синхронизация времени в случае недоступности хоста srv.linux.lab: <pre>server 127.127.1.0 fudge 127.127.1.0 stratum 10</pre> • Настроить системные параметры демона ntpd: <pre>driftfile /var/lib/ntp/drift logfile /var/log/ntp.log</pre> <p>4. Выполнить ручную синхронизацию времени с сервером времени srv.linux.lab, используя утилиту ntpdate с ключами -4 -d и -u.</p> <p>5. Запустить демон ntpd и добавить его в автозапуск на 3 и 5 уровнях выполнения, используя команды service и chkconfig.</p> <p>6. Проверить корректность синхронизации времени с сервером srv.linux.lab, используя команду ntpq:</p> <pre>remote refid st t when poll reach delay offset jitter ===== *srv.linux.lab LOCAL(0)11 u 26 64 377 0.378 3.978 2.957 LOCAL(0) .LOCL. 10 1 16 64 377 0.000 0.000 0.001</pre> <p>Если синхронизация времени настроена корректно, значение полей delay и offset должен быть больше нуля, а значение поля jitter не должно превышать 100 для локального сервера времени.</p>

Самостоятельные упражнения и дополнительные вопросы

1. Настроить динамическое обновление DNS записей с использованием цифровых подписей TSIG. В процессе выполнения данного задания рекомендуется пользоваться справочной документацией, находящейся в файле `/usr/share/doc/bind-<версия>/arm/Bv9ARM.html`, просмотреть которую можно при помощи текстового веб-браузера `lynx`.
2. Настроить MTA Sendmail на запрет отправки почты определенному пользователю, используя таблицу `access.db`. В процессе выполнения данного задания рекомендуется пользоваться справочной документацией, находящейся в файле `/usr/share/sendmail-cf/README`.
3. Настроить ограничение на подключение к FTP серверу для произвольно выбранного IP-адреса, используя при этом возможности демона `xinetd`. В процессе выполнения данного задания рекомендуется пользоваться примерной конфигурацией, находящейся в файле `/usr/share/doc/vsftpd-2.0.5/EXAMPLE/INTERNET_SITE`.
4. Опишите общую структуру файла формата LDIF. Какие данные можно хранить в каталоге LDAP и в каком виде они должны быть представлены в данном файле?
5. Какие типы виртуальных хостов можно организовать на веб-сервере Apache и чем они отличаются?

Практическая работа 3. Организация информационной безопасности ОС Linux

Цель

Цель работы – получение практических навыков организации информационной безопасности ОС Linux, таких как:

- настройка политики SELinux;
- настройка ограничений на доступ к сетевым сервисам при помощи демона `xinetd`;
- настройка межсетевого экрана на основе `iptables`.

Упражнение 3.1. Анализ работы механизма SELinux

Сценарий: В целях обеспечения большей информационной безопасности вы решили организовать на серверах, находящихся в демилитаризованной зоне, механизм SELinux. Вы не знаете, как включение данного механизма отразится на работе критичных сетевых сервисов, таких как веб-сервис и почтовый сервис. В связи с чем, вы решили для начала активировать механизм SELinux в тестовом режиме и проанализировать его работу при помощи утилиты **Setroubleshoot Browser**. Анализ работы механизма SELinux будет выполняться на виртуальной машине `srv.linux.lab`.

Задания	Описания действий
1. Настроить механизм SELinux на виртуальной машине <code>srv.linux.lab</code> .	1. Установить пакет audit и запустить демон auditd , используя команды chkconfig и service . 2. Проверить текущий режим работ SELinux, используя команду sestatus и активировать режим permissive в конфигурационном файле <code>/etc/sysconfig/selinux</code> . 3. Выполнить перезагрузку системы. 4. Выполнить повторный вход в систему, как пользователь root . 5. Проверить текущий режим работы SELinux, используя команду sestatus .
2. Проанализировать работу механизма SELinux , используя утилиту Setroubleshoot Browser .	1. Установить пакет setroubleshoot и policycoreutils-gui .

Задания	Описания действий
	<p>2. Запустить утилиту SElinux troubleshooter Browser, используя команду sealert -b (в графическом режиме).</p> <p>3. В меню программы SElinux troubleshooter Browser открыть вкладку Scan Logfile, указать путь к файлу /var/log/audit/audit.log и проанализировать системные сообщения, связанные с нарушением политики SELinux – Targeted. Необходимо проанализировать следующие параметры событий:</p> <ul style="list-style-type: none"> • Категория события (Category) _____; • Общая информация о событии (Summary) _____; • Рекомендации по изменению политики доступа (Allowing Access) _____.

Упражнение 3.2. Настройка ограничений на подключение к Telnet

Сценарий: Вы хотите организовать в вашей локальной сети сервис Telnet, причем, доступ к данному сервису необходимо предоставить только для определенных хостов. В данном упражнении вам предстоит реализовать задуманное на основе сервисов **xinetd** и **iptables**. В качестве сервера Telnet используется виртуальная машина **srv.linux.lab**, а в качестве Telnet клиента используется виртуальная машина **client.linux.lab**.

Задания	Описания действий
<p>1. Настроить сервис Telnet на виртуальной машине srv.linux.lab и ограничить сетевые подключения для указанных хостов.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Установить пакет telnet-server.</p> <p>3. Настроить конфигурационный файл /etc/xinetd.d/telnet следующим образом:</p> <pre> service telnet { flags = REUSE socket_type = stream wait = no user = root server = /usr/sbin/in.telnetd log_on_failure += USERID HOST ATTEMPT log_on_success += USERID DURATION TRAFFIC disable = no only_from = <IP-адрес> } </pre>

Задания	Описания действий
	<p>Здесь в директиве only_from необходимо указать IP-адрес хоста, с которого будет разрешен доступ по протоколу Telnet. Вместо параметра <IP-адрес> необходимо указать IP-адрес виртуальной машины client.linux.lab (Host-only).</p> <p>4. Указать в файле /etc/xinetd.d/telnet параметры запуска сервера telnet в режиме DEBUG (report) и используемый журнальный файл (/var/log/telnetd.log).</p> <p>5. Выполнить повторное считывание конфигурационного файла демона xinetd.</p>
<p>2. Проверить сделанные ограничения, используя клиент Telnet.</p>	<p>1. Подключиться к виртуальной машине srv.linux.lab с виртуальной машины client.linux.lab, используя команду telnet.</p> <p>2. Подключиться к серверу srv.linux.lab, используя команду telnet с локального IP-адреса 127.0.0.1 виртуальной машины srv.linux.lab.</p> <p>3. Проверить записи в журнальном файле /var/log/telnetd.log. Сделать соответствующие выводы:</p> <hr/> <hr/> <hr/>

Упражнение 3.3. Организация межсетевого экрана

Сценарий: Один из ваших серверов, на котором функционируют сервисы DNS, RSYNC, FTP и SSH, находится в демилитаризованной зоне (DMZ). Вам поставлена задача настроить межсетевой экран между периметром сети DMZ и внутренним периметром ЛВС. Межсетевой экран строится на основе IPTables и должен предоставлять доступ только к указанным сервисам. В качестве межсетевого экрана выступает виртуальная машина **srv.linux.lab**. В качестве клиента, находящегося во внутреннем периметре ЛВС, выступает виртуальная машина **client.linux.lab**.

Задания	Описания действий
<p>1. Определить открытые сетевые порты на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root.</p> <p>2. Установить пакет nmap.</p> <p>3. Выполнить сканирование портов виртуальной машины srv.linux.lab и определить открытые порты:</p> <pre># nmap 172.16.0.5 > /tmp/open_ports.1</pre> <p>4. Проанализировать файл /tmp/open_ports.1 и определить открытые порты _____</p>
<p>2. Настроить межсетевой экран на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root.</p> <p>2. Создать следующие правила IPTables:</p> <ul style="list-style-type: none"> • Разрешить весь сетевой трафик для локального интерфейса lo: <pre># iptables -A INPUT -i lo -j ACCEPT # iptables -A OUTPUT -o lo -j ACCEPT</pre> • Определить политики обработки пакетов по умолчанию: <pre># iptables --policy INPUT DROP # iptables --policy OUTPUT ACCEPT # iptables --policy FORWARD ACCEPT</pre> • Разрешить сетевой трафик для всех исходящих соединений и уже установленных входящих соединений: <pre># iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT # iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT</pre> • Разрешить ICMP протокол (ping, traceroute): <pre># iptables -A INPUT -i eth0 -p icmp --icmp-type destination-unreachable -j ACCEPT # iptables -A INPUT -i eth0 -p icmp --icmp-type time-exceeded -j ACCEPT # iptables -A INPUT -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT # iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request -j ACCEPT</pre> • Разрешить входящий сетевой трафик для сервисов DNS, RSYNC, FTP и SSH: <pre># iptables -A INPUT -p tcp -i eth0 --dport 53 -j ACCEPT # iptables -A INPUT -p tcp -i eth0 --dport 873 -j ACCEPT # iptables -A INPUT -p tcp -i eth0 --dport 21 -j ACCEPT # iptables -A INPUT -p tcp -i eth0 --dport 22 -j ACCEPT</pre> • Запретить весь остальной трафик не попадающий под предыдущие правила: <pre># iptables -A INPUT -j DROP</pre> <p>3. Просмотреть созданную политику, используя команду iptables -L и проверить, что настройки</p>

Задания	Описания действий
	<p>сделаны согласно поставленной задаче.</p> <p>4. Сохранить текущие настройки межсетевого экрана, используя команду service, и убедиться в сделанных изменениях, просмотрев файл /etc/sysconfig/iptables</p> <p>5. Выполнить команду очистки всех текущих политик IPTables, просмотреть текущие настройки и отключить их:</p> <pre># iptables --policy INPUT ACCEPT # iptables -F # iptables -L</pre> <p>6. Выполнить перезапуск сервиса IPTables и убедиться, что восстановились сохраненные ранее настройки:</p> <pre># service iptables restart # iptables -L</pre>
<p>3. Повторно определить открытые сетевые порты на виртуальной машине srv.linux.lab.</p>	<p>1. Выполнить сканирование портов виртуальной машины srv.linux.lab и определить открытые порты:</p> <pre># nmap 172.16.0.5 > /tmp/open_ports.2</pre> <p>2. Проанализировать файл /tmp/open_ports.2 и сделать соответствующие выводы _____</p> <p>_____</p> <p>_____</p>

Самостоятельные упражнения и дополнительные вопросы

1. Настройте межсетевой экран IPTables на виртуальной машине **srv.linux.lab** на журналирование пакетов, принятых по протоколу SSH.
2. Выполните одну из рекомендаций утилиты **SELinux troubleshooter Browser** и убедитесь, что события данного типа больше не возникают.
3. Опишите общий алгоритм работы механизма SELinux.
4. Какой командой можно просмотреть текущие параметры политики SELinux?
5. Настроить запуск демона **dovecot** с использованием **xinetd** и установить ограничение на количество одновременных соединений равное 2.

Упражнение 3.4. Организация контроля целостности файлов

Сценарий: В одном из филиалов вашей организации имеются незащищенные узлы, граничащие с публичной сетью Интернет и выполняющие роль пограничных маршрутизаторов. Вам поставлена задача организовать систему контроля изменений файлов. Данная система позволит администратору информационной безопасности определять попытки проникновения в систему, а также факты подмены критичных конфигурационных файлов.

В качестве основного системного ПО предлагается использовать локальную систему обнаружения вторжений AIDE.

Задания	Описания действий
1. Установить на сервере <code>srv.linux.lab</code> ПО AIDE.	1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root . 2. Установить пакеты gcc, ncurses-devel, zlib, bison, flex, curl . 3. Скопировать по FTP с виртуальной машины преподавателя архивы aide-0.14.tar.gz и mhash-0.9.9.9.tar.bz2 . 4. Распаковать архив mhash-0.9.9.9.tar.bz2 во временный каталог и выполнить сборку библиотек mhash из исходного кода, используя следующие команды: <pre># ./configure --prefix=/usr; make; make install</pre> 5. Распаковать архив aide-0.14.tar во временный каталог и выполнить сборку ПО AIDE из исходного кода, используя следующие команды: <pre># ./configure --prefix=/usr --with-zlib - # make && make install.</pre>
2. Настроить проверку основных конфигурационных файлов средствами AIDE	1. Создать конфигурационный файл aide.conf , скопировав его образец из каталога doc (архив с исходным кодом) в файл /etc/aide.conf . 2. Отредактировать файл aide.conf следующим образом: <ul style="list-style-type: none"> • Удалить все директивы @@ и !@@; • Установить параметры базы данных AIDE: <pre>database=file:/etc/aide.db database_out=file:/etc/aide.db.new</pre> • Определить правила сбора информации о файлах, добавив в конец файла /etc/aide.conf следующие строки: <pre>/etc p+u+g /boot All</pre>

Задания	Описания действий
	3. Выполнить начальную инициализацию базы данных AIDE: <pre># aide -c /etc/aide.conf --init # aide -c /etc/aide.conf --update # mv /etc/aide.db.new /etc/aide.db</pre> Сделать соответствующие выводы: _____ _____ _____
3. Смоделировать подмену нескольких файлов и выполнить проверку зафиксированных изменений	1. Изменить права на любом файле из каталога /etc 2. Изменить файл /boot/grub/grub.conf , добавив дополнительный параметр загрузки ядра audit . 3. Запустить команду проверки изменений: <pre># aide -c /etc/aide.conf --check</pre> Сделать соответствующие выводы: _____ _____ _____

Упражнение 3.5. Организация VPN туннеля

Сценарий: Вам необходимо предоставить доступ к внутренней сети предприятия для нескольких внешних сотрудников. Для защиты передаваемых данных предлагается организовать и протестировать защищенный туннель поверх публичной сети. Для решения данной задачи вам необходимо развернуть VPN-сервер и протестировать подключение VPN-клиентом. В качестве VPN-сервера выступит виртуальная машина **srv.linux.lab**, VPN-клиентом является виртуальная машина **client.linux.lab**.

В качестве основного системного ПО предлагается использовать OpenVPN.

Задания	Описания действий
1. Установить на виртуальной машине srv.linux.lab ПО OpenVPN.	1. Выполнить вход в систему на виртуальной машине srv.linux.lab под пользователем root . 2. Установить пакет openvpn , используя менеджер пакетов yum .
2. Установить на виртуальной машине client.linux.lab ПО OpenVPN.	1. Выполнить вход в систему на виртуальной машине client.linux.lab под пользователем root . 2. Установить пакет openvpn , используя менеджер пакетов yum .

Задания	Описания действий
<p>3. Настроить ПО OpenVPN на виртуальных машинах srv.linux.lab и client.linux.lab</p>	<p>1. Создать инфраструктуру PKI, выполнив следующие действия:</p> <ul style="list-style-type: none"> • Создать удостоверяющий центр сертификатов: <pre># cp /usr/share/doc/openvpn/examples/easy-rsa/2.0/ /etc/openvpn/easy-rsa/2.0 # cd /etc/openvpn/easy-rsa/2.0</pre> • Отредактировать файл /etc/openvpn/easy-rsa/2.0/vars следующим образом: <pre>export KEY_SIZE=2048 export KEY_COUNTRY=RU export KEY_PROVINCE=NA export KEY_CITY=Moscow export KEY_ORG="SOFTLINE" export KEY_EMAIL="root@localhost"</pre> • Выполнить следующие команды: <pre>#. ./vars #./clean-all #./build-ca</pre> • Создать сертификат для VPN-сервера: <pre># ./build-key-server <FQDN-имя VPN-сервера></pre> • Сгенерировать сертификат для клиента: <pre># ./build-key-server <FQDN-имя VPN-клиента></pre> • Сгенерировать параметры шифрования Diffie-Hellman и создать ключ аутентификации TLS: <pre># ./build-dh # cd keys/ # openvpn --genkey --secret ta.key</pre> • Создать катлог для хранения ключей и сертификатов и поместить в него нужные файлы: <pre># mkdir -m 0700 /etc/openvpn/keys # cp ca.crt ../../keys # mv dh2048.pem ta.key <Имя VPN-сервера>.crt <Имя VPN-сервера>.key ../../keys</pre> <p>2. Скопировать в каталог /etc/openvpn/keys на клиенте следующие файлы:</p> <pre><FQDN-имя VPN-клиента>.key <FQDN-имя VPN-клиента>.crt ta.key ca.crt</pre> <p>3. Создать конфигурационный файл server.conf для VPN-сервера и поместить его в каталог /etc/openvpn:</p> <pre>## server.conf local <Host-only IP-адрес VPN-сервера> port 1194 proto udp dev tun daemon</pre>

Задания	Описания действий
	<pre>server 10.0.0.0 255.255.255.0 max-clients 25 ca /etc/openvpn/keys/ca.crt cert /etc/openvpn/keys/<Имя VPN-сервера>.crt key /etc/openvpn/keys/<Имя VPN-сервера>.key dh /etc/openvpn/keys/dh1024.pem tls-auth /etc/openvpn/keys/ta.key 0 cipher BF-CBC comp-lzo keepalive 10 120 log-append /var/log/openvpn.log status /var/log/openvpn-status.log ifconfig-pool-persist /etc/openvpn/ipp.txt mute 20 verb 4</pre> <p>4. Создать конфигурационный файл client.ovpn для VPN-клиента и поместить его в каталог /etc/openvpn/config:</p> <pre>## client.conf client pull dev tun proto udp remote <Host-only IP-адрес VPN-сервера> 1194 ca /etc/openvpn/keys/ca.crt cert /etc/openvpn/keys/<Имя VPN-сервера>.crt key /etc/openvpn/keys/<Имя VPN-сервера>.key tls-auth /etc/openvpn/keys/ta.key 1 cipher BF-CBC comp-lzo verb 4 mute 20 ns-cert-type server</pre> <p>5. Запустить процесс OpenVPN на сервере:</p> <pre># openvpn /etc/openvpn/server.conf</pre>
<p>4. Выполните тестовое подключение VPN-клиентом к VPN-серверу</p>	<p>1. Запустить процесс OpenVPN на клиенте. 2. Выполните команду ping для частных IP-адресов клиента и сервера и убедитесь, что сервер доступен для клиента. Сделайте соответствующие выводы:</p> <hr/> <hr/> <hr/>