МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

АДМИНИСТРИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета

Составитель **Г.Е. Шевелёв**

Издательство
Томского политехнического университета
2019

УДК 519.8
ББК 22.17я73
Т24

Т24

Администрирование вычислительных систем и сетей: учебное пособие / сост. Г.Е. Шевелев; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2019. – 159 с.
В авторской редакции

Учебное пособие состоит из восьми глав, написано в соответствии с программой курса «Администрирование вычислительных систем и сетей» для вузов. В нем дан обзор современных вычислительных систем и сетевых технологий, рассмотрены основные задачи и функции, выполняемые системным администратором по созданию и обслуживанию вычислительных систем и сетей.

Пособие подготовлено в отделении информационных технологий ИШИТР ТПУ и предназначено для студентов ИШИТР, обучающихся по направлению 01.03.02.

УДК 519.8
ББК 22.17я73

Рецензенты
Доктор физико-математических наук
Профессор ТГАСУ
Б.М. Шумилов

Кандидат технических наук
доцент ТУСУР
А.А. Шелестов

Оглавление

Введение 7
1. Вычислительные системы 9
Понятие вычислительной системы 9
Классификация вычислительных систем 9
Суперкомпьютеры и особенности их архитектуры 13
Кластерные суперкомпьютеры и особенности их архитектуры 14
Квантовый компьютер 19
2. Использование виртуальных машин 21
Понятие виртуальной машины 21
Понятие виртуализации 26
Место и роль виртуальных машин в образовательном процессе 30
Обзор наиболее известных виртуальных машин 32
3. Введение в сетевые технологии 36
Локальные сети 36
Глобальные сети 37
Архитектура сети 38
Общая шина 38
Звезда 38
Кольцо 39
Сетевой уровень и маршрутизация 39
Адресация: сеть и хост-машина 40
Маршрутизация с использованием сетевых адресов 41
Протоколы маршрутизации и маршрутизируемые протоколы 41
Статические и динамические маршруты 42
Эталонная модель OSI 42
Модель OSI 43
Уровень приложений 44
Уровень представлений 44
Уровень сеансовый 44
Уровень транспортный 45
Уровень сетевой 45
Уровень канальный 45

Уровень физический 45
Инкапсулирование данных 46
Сетевые стандарты Ethernet и IEEE 802.3 48
Принцип работы сеть Ethernet 802.3 49
Множественный доступ с контролем несущей и обнаружением конфликтов 49
IP-адресация 51
Классы IP-адресов 52
Зарезервированные классы сетей 52
Адреса класса A 53
Адреса класса B 54
Адреса класса C 54
Выделение подсетей 54
4. Сетевое администрирование **55**
Управление компьютерной сетью 55
Системное и сетевое администрирование 56
Цели и задачи администратора сети 58
Автоматизация управления сетью 59
Многопользовательские информационные системы 61
Многопользовательские объектно-ориентированные среды 62
Особенности работы в многопользовательских средах 63
Различные сетевые операционные системы и особенности администрирования в них 65
Сравнение сетевых ОС 75
Администрирование в сетях с операционными системами типа Windows 77
Администрирование в среде Unix/Linux 80
Назначение и функционирование брандмауэра 80
5. Виртуальные частные сети **82**
Определение виртуальных частных сетей 82
Развертывание пользовательских виртуальных частных сетей 84
Преимущества пользовательских VPN 85
Проблемы, связанные с пользовательскими VPN 86
Управление пользовательскими VPN 87
Развертывание узловых сетей VPN 88
Преимущества узловых VPN 88

Проблемы, связанные с узловыми VPN 89
Управление узловыми VPN 90
Понятие стандартных технологий функционирования VPN 90
Сервер VPN 91
Алгоритмы шифрования 93
Система аутентификации 94
Протокол VPN 94
Обзор существующих VPN-сервисов 95
Прокси-сервер 97
Как работает прокси-соединение 98
Зачем и кому нужен прокси 98
Преимущества 99
Виды прокси-серверов 99
Когда использовать прокси-сервер, а когда VPN? 101
Базовое руководство по установке и настройке прокси-серверов 103
6. Основы администрирования *Linux* **106**
Основные задачи системного администрирования 106
Удаленный доступ к серверу Linux 109
Диагностика сети Linux 110
Мониторинг ресурсов системы 112
Проверка работоспособности серверов 114
Просмотр логов 115
Установка программного обеспечения 116
7. Системное администрирование с помощью сетевых инструментов *Linux* **117**
Утилиты администратора 117
Ping 117
Traceroute 118
Telnet 118
Netstat 118
Nmcli 120
Маршрутизация 121
Tcpdump и Wireshark 123
Iptables 125

Nslookup *125*
Поиск неполадок *126*
8. Управление учётными записями пользователей и доступом к ресурсам ……...127
Учётные записи пользователей *127*
Имя пользователя *127*
Соглашения об именовании *127*
Пароли *130*
Слабые пароли *131*
Сильные пароли *133*
Срок действия пароля *134*
Информация об управлении доступом *135*
Повседневное управление учётными записями и доступом к ресурсам *135*
Новые сотрудники *135*
Прекращение работы *136*
Переход на другую должность *137*
Управление ресурсами пользователей *138*
Кому разрешён доступ к общим данным? *138*
Общие группы и данные *138*
Определение структуры групп *139*
Куда пользователи обращаются за общими данными? *139*
Вопросы глобального владения *139*
Домашние каталоги *140*
Каким образом предотвращается нецелевое использование ресурсов? *140*
Приложение *141*

Введение

Системный администратор – сотрудник, должностные обязанности которого подразумевают обеспечение штатной работы парка компьютерной техники, сети и программного обеспечения, а также обеспечение информационной безопасности в организации. Разговорные названия – сисадми́н (англ. *sysadmin*), нередко просто админ.

Системные администраторы – сотрудники, в обязанности которых входит не только слежение за сетевой безопасностью организации, но и создание оптимальной работоспособности компьютеров и программного обеспечения для пользователей, часто связанных между собой общей работой на определенный результат.

Нередко функции системного администратора перекладывают на компании, занимающиеся *IT*-аутсорсингом. Обычно такие компании предоставляют более низкую, чем содержание штатного сотрудника, стоимость обслуживания и осуществляют работу на основе абонементных договоров.

Ввиду быстрого роста Интернета и развития сетевых технологий, системному администратору-одиночке становится всё сложнее противостоять всем проблемам, поэтому давно появились специализированные форумы и печатные издания, направленные на расширение кругозора начинающих системных администраторов и оказание помощи в решении различных проблем.

Профессиональный праздник системного администратора – последняя пятница июля.

В круг типовых задач системного администратора обычно входит:

- подготовка и сохранение резервных копий данных, их периодическая проверка и уничтожение; установка и конфигурирование необходимых обновлений для операционной системы и используемых программ;
- установка и конфигурирование нового аппаратного и программного обеспечения;
- создание и поддержание в актуальном состоянии пользовательских учётных записей;
- ответственность за информационную безопасность в компании;
- устранение неполадок в системе;
- планирование и проведение работ по расширению сетевой структуры предприятия;
- документирование всех произведенных действий.

В организациях с большим штатом сотрудников данные обязанности могут делиться между несколькими системными администраторами – например, между администраторами безопасности, учётных записей и резервного копирования. Также, в организациях с небольшим штатом сотрудников эти обязанности могут исполняться одним специалистом, занимающимся как консультированием пользователей, так и ремонтом аппаратной части персональных компьютеров и периферийных устройств.

Системных администраторов можно разделить на несколько категорий:

1. ***Администратор веб-сервера*** – занимается установкой, настройкой и обслуживанием программного обеспечения веб-серверов. Как правило, работает в хостинговой компании.

Необходимы знания *Unix*-систем (главным образом *Linux* и *FreeBSD*), умение конфигурировать веб-сервер *Apache* и почтовые сервера (*qmail, Sendmail, Exim, Postfix*), которые установлены на более чем 90 % *web*-серверов во всем мире; дополнительно веб-сервер *IIS* и ОС семейства *Windows Server*. Обязательно понимание модели *OSI*, глубокие познания в области сетевых протоколов (стек *TCP/IP, IPX*) и их реализации, маршрутизации, реализации *VPN (Virtual Private Network* «виртуальная частная сеть») — обобщённое название технологий, позволяющих обеспечить одно или несколько сетевых соединений (логическую сеть) поверх другой сети (например, Интернет)*)*, физическом построении сетей (*Ethernet, Token ring, FDDI*, 802.11).

2. ***Администратор базы данных*** – специализируется на обслуживании баз данных.

Нужны глубокие знания СУБД (как минимум одной из *MySQL, PostgreSQL, MS SQL, Oracle, Informix, Firebird*), операционной системы, на которой работает база данных (*Windows Server*, *nix (главным образом *Linux/FreeBSD*) или *Solaris*), знание особенностей реализации баз данных, а также знание информационно-логического языка *SQL*.

1. Вычислительные системы

Понятие вычислительной системы

В связи с кризисом классической структуры ЭВМ дальнейшее поступательное развитие вычислительной техники напрямую связано с переходом к параллельным вычислениям, с идеями построения многопроцессорных систем и сетей, объединяющих большое количество отдельных процессоров и (или) ЭВМ. Здесь появляются огромные возможности совершенствования средств вычислительной техники. Но следует отметить, что при несомненных практических достижениях в области параллельных вычислений, до настоящего времени отсутствует их единая теоретическая база.

Термин ***вычислительная система*** появился в начале - середине 60-х гг. при появлении ЭВМ III поколения. Это время знаменовалось переходом на новую элементную базу – интегральные схемы. Следствием этого явилось появление новых технических решений: разделение процессов обработки информации и ее ввода-вывода, множественный доступ и коллективное использование вычислительных ресурсов в пространстве и во времени. Появились сложные режимы работы ЭВМ – *многопользовательская* и *многопрограммная обработка.*

Под ***вычислительной системой*** (ВС) понимают совокупность взаимосвязанных и взаимодействующих процессоров или ЭВМ, периферийного оборудования и программного обеспечения, предназначенную для сбора, хранения, обработки и распределения информации.

Отличительной особенностью ВС по отношению к ЭВМ является наличие в них нескольких вычислителей, реализующих параллельную обработку. Создание ВС преследует следующие основные цели: повышение производительности системы за счет ускорения процессов обработки данных, повышение надежности и достоверности вычислений, предоставление пользователям дополнительных сервисных услуг и т.д.

Классификация вычислительных систем

Существует большое количество признаков, по которым классифицируют вычислительные системы.

По назначению вычислительные системы делят на:

- универсальные;
- специализированные.

Специализированные системы ориентированы на решение узкого класса задач, в отличие от универсальных, предназначенных для широкого спектра задач.

По типу построения вычислительные системы разделяются на:

- многомашинные;
- многопроцессорные.

Это значит, что вычислительные системы могут строиться на базе нескольких компьютеров или на базе нескольких процессоров. В первом случая ВС будет *многомашинной,* во втором – *многопроцессорной.*

Многомашинные вычислительные системы (ММС) появились раньше, чем многопроцессорные. Основные отличия ММС заключаются в организации связей и обмена информацией между ЭВМ комплекса. Многомашинная *ВС* содержит некоторое число компьютеров, информационно взаимодействующих между собой. Машины могут находиться рядом друг с другом, а могут быть удалены друг от друга на некоторое, иногда значительное расстояние (вычислительные сети). В многомашинных *ВС* каждый компьютер работает под управлением своей операционной системы (ОС). А поскольку обмен информацией между машинами выполняется под управлением ОС, взаимодействующих друг с другом, динамические характеристики процедур обмена несколько ухудшаются (требуется время на согласование работы самих ОС). Информационное взаимодействие компьютеров в многомашинной ВС может быть организовано на уровне:

1. процессоров;
2. оперативной памяти;
3. каналов связи.

При непосредственном взаимодействии процессоров друг с другом информационная связь реализуется через регистры процессорной памяти и требует наличия в ОС весьма сложных специальных программ.

Взаимодействие на уровне оперативной памяти (ОП) сводится к программной реализации общего поля оперативной памяти, что несколько проще, но также требует существенной модификации ОС. Под общим полем имеется в виду равнодоступность модулей памяти: все модули памяти доступны всем процессорам и каналам связи.

На уровне каналов связи взаимодействие организуется наиболее просто и может быть достигнуто внешними по отношению к ОС программами-драйверами, обеспечивающими доступ от каналов связи одной машины к внешним устройствам других (формируется общее поле внешней памяти и общий доступ к устройствам ввода-вывода).

Ввиду сложности организации информационного взаимодействия на 1-м и 2-м уровнях в большинстве многомашинных ВС используется 3-й уровень, хотя и динамические характеристики (в первую очередь быстродействие), и показатели надежности таких систем существенно ниже.

Многопроцессорные системы (МПС) содержат несколько процессоров, информационно взаимодействующих между собой либо на уровне регистров процессорной памяти, либо на уровне ОП. Этот тип взаимодействия используется в большинстве случаев, ибо организуется значительно проще и сводится к созданию общего поля оперативной памяти для всех процессоров. Общий доступ к внешней памяти и устройствам ввода-вывода обеспечивается обычно через каналы ОП. Важным является и то, что многопроцессорная вычислительная система

работает под управлением единой ОС, общей для всех процессоров. Это существенно улучшает динамические характеристики ВС, но требует наличия специальной, весьма сложной ОС.

Однако МПС имеют и существенные недостатки. Они, в первую очередь, связаны с использованием ресурсов общей оперативной памяти. При большом количестве объединяемых процессоров возможно возникновение конфликтных ситуаций, в которых несколько процессоров обращаются с операциями типа *чтение* и *запись* к одним и тем же ячейкам памяти. Помимо процессоров к ОП подключаются все процессоры ввода-вывода, средства измерения времени и т.д. Поэтому вторым серьезным недостатком МПС является проблема коммутации и доступа абонентов к ОП. Процедуры взаимодействия очень сильно усложняют структуру ОС МПС. Опыт построения подобных систем показал, что они эффективны при небольшом числе объединяемых процессоров (от 2 до 10). Типичным примером массовых многомашинных ВС могут служить компьютерные сети, примером многопроцессорных ВС – *суперкомпьютеры.*

По типу ЭВМ или процессоров, используемых для построения ВС, различают:

- однородные системы
- неоднородные системы.

Однородная ВС строится на базе однотипных компьютеров или процессоров. Однородные системы позволяют использовать стандартные наборы технических, программных средств, стандартные протоколы (процедуры) сопряжения устройств. Поэтому их организация значительно проще, облегчается обслуживание систем и их модернизация.

Неоднородная ВС включает в свой состав различные типы компьютеров или процессоров. При построении системы приходится учитывать их различные технические и функциональные характеристики, что существенно усложняет создание и обслуживание неоднородных систем.

По методам управления элементами ВС различают

- централизованные;
- децентрализованные;
- со смешанным управлением.

В централизованных ВС за управление отвечает главная или диспетчерская ЭВМ (процессор). Ее задачей является распределение нагрузки между элементами, выделение ресурсов, контроль состояния ресурсов, координация взаимодействия. Централизованный орган управления в системе может быть жестко фиксирован или эти функции могут передаваться другой ЭВМ (процессору), что способствует повышению надежности системы. Централизованные системы имеют более простые ОС.

В децентрализованных системах функции управления распределены между ее элементами. Каждая ЭВМ (процессор) системы сохраняет известную автономию, а необходимое взаимодействие между элементами устанавливается по

специальным наборам сигналов. С развитием ВС и, в частности, сетей ЭВМ, интерес к децентрализованным системам постоянно растет.

В ***системах со смешанным управлением*** совмещаются процедуры централизованного и децентрализованного управления. Перераспределение функций осуществляется в ходе вычислительного процесса, исходя из сложившейся ситуации.

***По принципу закрепления вычислительных* функций** за отдельными ЭВМ (процессорами) различают системы с *жестким* и *плавающим* закреплением функций. В зависимости от типа ВС следует решать задачи статического или динамического размещения программных модулей и массивов данных, обеспечивая необходимую гибкость системы и надежность ее функционирования.

По степени территориальной разобщенности вычислительных модулей ВС делятся на системы:

- ***территориально-сосредоточенные*** – это когда все компоненты располагаются в непосредственной близости друг от друга;
- ***распределенные*** – это когда компоненты могут располагаться на значительном расстоянии, например, вычислительные сети;
- ***структурно-одноуровневые*** – это когда имеется лишь один общий уровень обработки данных;
- ***многоуровневые*** (иерархические) структуры – это когда в иерархических ВС машины или процессоры распределены по разным уровням обработки информации, некоторые машины (процессоры) могут специализироваться на выполнении определенных функций.

По режиму работы ВС различают системы, работающие **в** *оперативном* и *неоперативном* временных режимах. Первые, как правило, используют режим реального масштаба времени. Этот режим характеризуется жесткими ограничениями на время решения задач в системе и предполагает высокую степень автоматизации процедур ввода-вывода и обработки данных.

На рис. 1 представлена принципиальная схема классификации вычислительных систем.

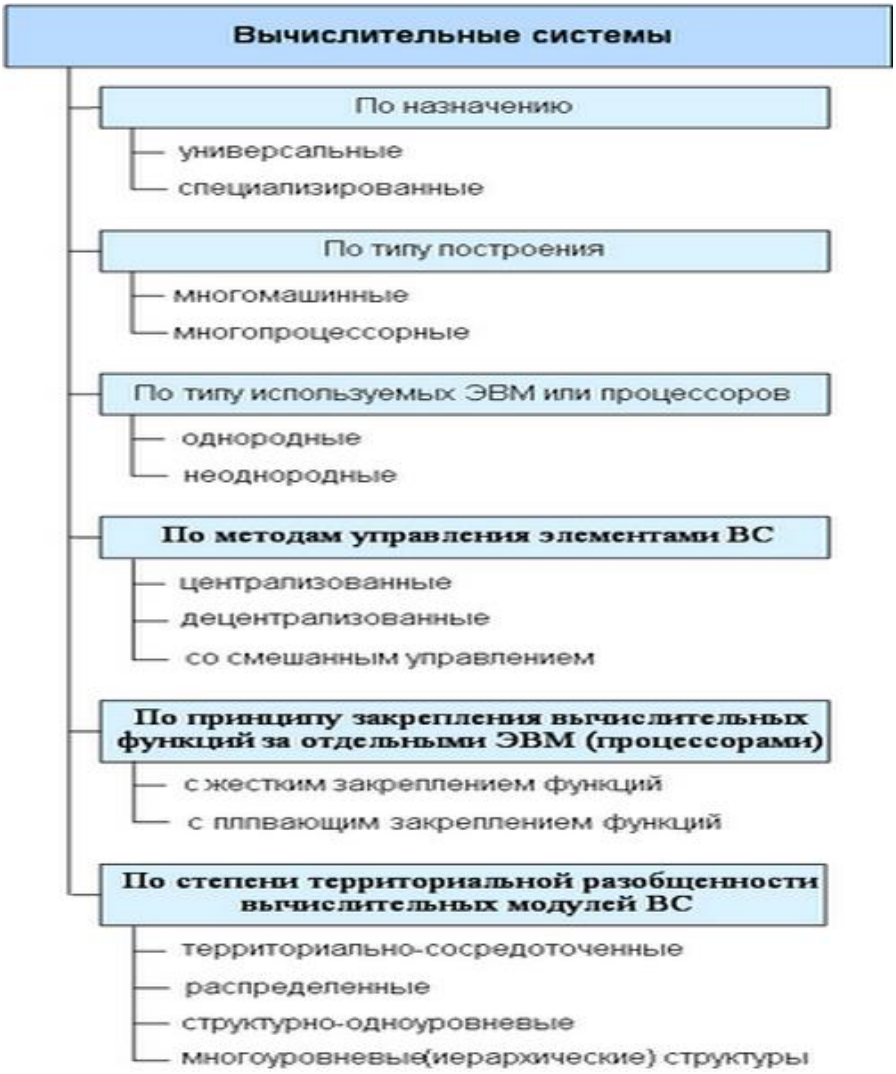


Рис. 1. Принципиальная схема классификации вычислительных систем

Суперкомпьютеры и особенности их архитектуры

К суперкомпьютерам относятся мощные многопроцессорные вычислительные машины с быстродействием сотни миллионов – десятки миллиардов операций в секунду. Создать такие высокопроизводительные компьютеры на одном микропроцессоре (МП) не представляется возможным ввиду ограничения, обусловленного конечным значением скорости распространения электромагнитных волн (300 000 км/с), т.к. время распространения сигнала на расстояние несколько миллиметров (линейный размер стороны МП) при быстродействии 100 млрд операций/с становится соизмеримым со временем выполнения одной операции. Поэтому суперкомпьютеры создаются в виде высокопараллельных многопроцессорных вычислительных систем (МПВС).

Высокопараллельные МПВС имеют несколько разновидностей:

1. ***Магистральные*** (конвейерные) МПВС, у которых процессор одновременно выполняет разные операции над последовательным потоком обрабатываемых данных. По принятой классификации такие МПВС относятся к

системам с многократным потоком команд и однократным потоком данных (МКОД или MISD – *Multiple Instruction Single Data*).

2. ***Векторные*** МПВС, у которых все процессоры одновременно выполняют одну команду над различными данными – однократный поток команд с многократным потоком данных (ОКМД или SIMD – *Single Instruction Multiple Data*).

3. ***Матричные*** МПВС, у которых микропроцессор одновременно выполняет разные операции над последовательными потоками обрабатываемых данных — многократный поток команд с многократным потоком данных (МКМД или MIMD – *Multiple Instruction Multiple Data*).

В суперкомпьютере используются все три варианта архитектуры МПВС:

- структура *MIMD* в классическом ее варианте;
- параллельно-конвейерная модификация, иначе *MMISD*, то есть многопроцессорная *(Multiple*) *MISD*-архитектура;
- параллельно-векторная модификация, иначе MSIMD, то есть многопроцессорная SIMD-архитектура.

Первый суперкомпьютер был задуман в 1960 и создан в 1972 году (машина *ILLIAC* IV с производительностью 20 *MFloPS*), а начиная с 1974 года лидерство в разработке суперкомпьютеров захватила фирма *Cray Research*, выпустившая *Cray* 1 производительностью 160 *MFloPS* и объемом оперативной памяти 64 Мбайт.

Англоязычный термин *MFLOPS* является сокращением ***M****illions of* ***FL****oating point* ***OP****erations per* ***S****econd* и означает Миллион операций с плавающей точкой в секунду. *MFLOPS* употребляется изготовителями высокопроизводительных вычислительных систем для указания их вычислительной мощности при операциях с числами с плавающей точкой.

Кластерные суперкомпьютеры и особенности их архитектуры

Существует технология построения больших компьютеров и суперкомпьютеров на базе кластерных решений. По мнению многих специалистов, на смену отдельным, независимым суперкомпьютерам должны прийти группы высокопроизводительных серверов, объединяемых в кластер.

Кластер – это связанный набор полноценных компьютеров, используемый в качестве единого вычислительного ресурса.

Удобство построения кластерных ВС заключается в том, что можно гибко регулировать необходимую производительность системы, подключая к кластеру с помощью специальных аппаратных и программных интерфейсов обычные серийные серверы до тех пор, пока не будет получен суперкомпьютер требуемой мощности. Кластеризация позволяет манипулировать группой серверов как одной системой, упрощая управление и повышая надежность.

Важной особенностью кластеров является обеспечение доступа любого сервера к любому блоку как оперативной, так и дисковой памяти. Эта проблема успешно решается, например, объединением систем *SMP*-архитектуры на базе автономных серверов для организации общего поля оперативной памяти и

использованием дисковых систем *RAID* для памяти внешней (*SMP – Shared Memory multiprocessing*, технология мультипроцессирования с разделением памяти).

Симметричная многопроцессорность (*Symmetric Multiprocessing*, сокращённо ***SMP***) — архитектура многопроцессорных компьютеров, в которой два или более одинаковых процессора сравнимой производительности подключаются единообразно к общей памяти (и периферийным устройствам) и выполняют одни и те же функции.

RAID — Redundant Array of Independent Disks («избыточный (резервный) массив *независимых* дисков»)

Для создания кластеров обычно используются либо простые однопроцессорные персональные компьютеры, либо двух- или четырехпроцессорные *SMP*-серверы. При этом не накладывается никаких ограничений на состав и архитектуру узлов. Каждый из узлов может функционировать под управлением своей собственной операционной системы. Чаще всего используются стандартные ОС: *Linux, FreeBSD, Solaris, Unix, Windows NT*. В тех случаях, когда узлы кластера неоднородны, то говорят о ***гетерогенных*** кластерах.

При создании кластеров можно выделить два подхода:

1. Первый подход применяется при создании небольших кластерных систем. В кластер объединяются полнофункциональные компьютеры, которые продолжают работать и как самостоятельные единицы, например, компьютеры учебного класса или рабочие станции лаборатории.

2. Второй подход применяется в тех случаях, когда целенаправленно создается мощный вычислительный ресурс. Тогда системные блоки компьютеров компактно размещаются в специальных стойках, а для управления системой и для запуска задач выделяется один или несколько полнофункциональных компьютеров, называемых хост-компьютерами. В этом случае нет необходимости снабжать компьютеры вычислительных узлов графическими картами, мониторами, дисковыми накопителями и другим периферийным оборудованием, что значительно удешевляет стоимость системы.

Основные достоинства кластерных суперкомпьютерных систем:

- высокая суммарная производительность;
- высокая надежность работы системы;
- наилучшее соотношение производительность/стоимость;
- возможность динамического перераспределения нагрузок между серверами;
- легкая масштабируемость, то есть наращивание вычислительной мощности путем подключения дополнительных серверов;
- удобство управления и контроля работы системы.

Наряду с достоинствами, как и у любой технологии, у кластеризации имеются свои недостатки:

- задержки разработки и принятия общих стандартов;
- большая доля нестандартных и закрытых разработок различных фирм, затрудняющих их совместное использование;
- трудности управления одновременным доступом к файлам;

- сложности с управлением конфигурацией, настройкой, развертыванием, оповещениями серверов о сбоях и т.п.

Самый мощный в мире компьютер

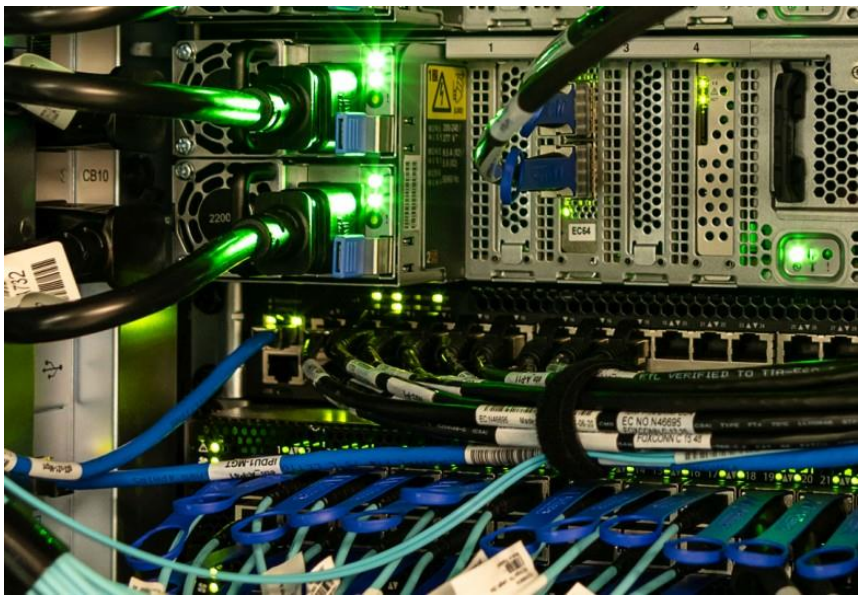
Самый мощный в мире компьютер был создан недавно. В 2018 году американские учёные из лаборатории Оук-Ридж показали миру устройство *Summit*, произведённое компанией *IBM*. Суперкомпьютер является самым производительным вычислительным комплексом в мире на сегодняшний день.

ОК-Риджская национальная лаборатория (*Oak Ridge National Laboratory; ORNL* – крупнейшее научно-исследовательское учреждение в системе национальных лабораторий Министерства энергетики США, расположена вблизи города Ок-Ридж (штат Теннесси) рядом с городом Ноксвилл. Научные направления: материаловедение, нейтронная физика, энергетика, высокопроизводительные вычисления, системная биология, национальная безопасность.

Мощнейшая машина чем-то похожа на привычный домашний ПК: здесь тоже есть корпус, за которым расположились важные компоненты. Правда, чтобы спрятать всю вычислительную мощь, разработчикам пришлось создать несколько огромных коробок, занявших площадь 860 м².

Комплекс больше напоминает ряды книжных полок в библиотеке, на которых располагаются многочисленные узлы, коих насчитывается 4608 штук. В каждом размещены комплектующие, отвечающие за вычислительные способности суперкомпьютера. Для соединения элементов системными администраторами было проложено порядка 300 км оптоволокна.

При работе всех компонентов устройства нахождение внутри комплекса может быть опасным для слуха. Работники лаборатории заходят в помещение с суперкомпьютером только в наушниках.



Характеристики суперкомпьютера

Summit располагает 4608 узлами внутри корпуса. Каждый содержит высокопроизводительные элементы, среди которых два 22-ядерных процессора от *IBM* (в общей сложности 2.28 млн ядер на всю систему), а также шесть видеокарт *Nvidia Tesla V100* (цена за единицу достигает 600 тыс. рублей). Объединяются узлы высокоскоростной сетью, способной пересылать данные между серверами при пропускной способности 200 Гбит/с. Внутри каждого узла установлен чип гибридной оперативной памяти на 500 Гб.

Совместная работа серверных видеокарт *Nvidia Tesla V100* при полной нагрузке способна выдать показатели производительности в районе 3,3 эксафлопс, то есть 3×10^{18}операций с плавающей запятой в секунду.

Громадина, установленная в лаборатории Оук-Ридж, весит безумные 340 тонн! Для сравнения, столько же весит самолёт Боинг 747.

Summit оказывается и весьма прожорливым устройством, которому для нормального функционирования требуется 15 МВт энергии. Такого количества хватило бы, чтобы запитать 8100 жилых домов.

Внутренние компоненты суперкомпьютера очень нагреваются при высоких нагрузках, поэтому для их охлаждения используется продвинутая водяная система, перегоняющая порядка 18 тыс. литров воды в минуту.

Работники лаборатории считают, что высокие мощности нового суперкомпьютера позволят продвинуться в работе над обучением искусственного интеллекта.

Также *Summit* призван проводить сложные многоступенчатые расчёты, которые позволят выявлять генетические предрасположенности к болезням и аллергенам.

Сопоставить многочисленные вариации генетических кодов способен только суперкомпьютер. *Summit* справится с этой задачей лучше всех.

В расчётах суперкомпьютера заинтересованы представители министерства энергетики США, считающие, что устройство поможет в работе с термоядерными разработками и изучением альтернативных видов энергии.

Суперкомпьютер способен провести расчёты в планировании и создании термоядерных механизмов.

Учёные метеорологи уже нашли применение *Summit*, разработав программу более точного предсказания погоды и климатического моделирования.

Мгновенный анализ сейсмологических данных позволит быстрее предсказывать погоду или предупреждать о катаклизмах.

Возможности *Summit* безграничны, а мощность позволит шагнуть технологиям далеко в изучении и развитии многочисленных научных и общественных сфер.

В России в 2009 году в МГУ установили суперкомпьютер «Ломоносов», который содержит 6 654 вычислительных узла, более 94 000 процессорных ядер, обладает пиковой производительностью 1,37 Пфлопс. Реальная производительность системы на тесте Linpack равна 674 Тфлопс, что позволило ему занять в июне 2011 года 13–е место в списке Top500 самых производительных компьютеров мира.

FLOPS (***FL****oating-point* ***O****perations* ***P****er* ***S****econd*) – внесистемная единица, используемая для измерения производительности компьютеров, показывающая, сколько операций с плавающей запятой в секунду выполняет данная вычислительная система.

Терафлопс = =10^{12} Флопс

Петафлопс = 10^{15} Флопс

Эксафлопс = 10^{18} Флопс

В марте 2018 года в МГУ объявили, что закончена разработка и началась опытная эксплуатация нового раздела суперкомпьютера «Ломоносов-2» с пиковой производительностью 1,8 Пфлопс, что в будущем позволит ему подойти вплотную

к рубежу в 5 Пфлопс. Новый раздел суперкомпьютера базируется на многоядерных процессорах *Intel Skylake* и графических процессорах *NVidia Pascal P100* с аппаратной поддержкой высокоскоростного интерфейса *NVLink.*

Российские ученые считают, что отставание России от США и Китая в петафлопсах не так страшно. Ключ к эффективному использованию машин – высокий математический уровень и класс специалистов в области программного обеспечения

Квантовый компьютер

Российские ученые представили разработку, которая, по их словам, должна кардинально изменить жизнь человечества. Созданием квантовых компьютеров, способных работать в миллионы раз быстрее современных операционных систем, занимаются крупнейшие технологические корпорации мира. Но они уже признали победу коллег.

Это казалось фантастикой еще вчера – квантовые компьютеры, способные обогнать все существующие устройства. Они настолько мощные, что могут или открыть человечеству новые горизонты, или обрушить все системы безопасности, потому что смогут взломать их. Расчеты, которые на сегодняшнем суперкомпьютере займут тысячи лет, квантовый может сделать в один миг.

Российские ученые представили разработку, которая, по их словам, должна кардинально изменить жизнь человечества. Созданием квантовых компьютеров, способных работать в миллионы раз быстрее современных операционных систем, занимаются крупнейшие технологические корпорации мира. Но они уже признали победу коллег.

Это казалось фантастикой еще вчера – квантовые компьютеры, способные обогнать все существующие устройства. Они настолько мощные, что могут или открыть человечеству новые горизонты, или обрушить все системы безопасности, потому что смогут взломать их.

В разработку вкладываются крупнейшие корпорации: *Google, IBM, Microsoft, Alibaba*. Но сегодня в центре внимания – Михаил Лукин, физик из Гарварда и один из основателей Российского квантового центра. Его команде удалось создать самый мощный на данный момент квантовый компьютер.

«Это одна из самых больших квантовых систем, которые были созданы. Мы входим в тот режим, где уже классические компьютеры не могут справится с вычислениями. Делаем маленькие открытия уже, увидели новые эффекты, которые не ожидались теоретически, которые мы сейчас можем, мы пытаемся понять, мы даже до конца их не понимаем», – рассказывает профессор Гарвардского университета, сооснователь Российского квантового центра Михаил Лукин.

Все – из-за мощности таких устройств. Расчеты, которые на сегодняшнем суперкомпьютере займут тысячи лет, квантовый может сделать в один миг.

Как это работает? В обычных компьютерах информация и вычисления – это биты. Каждый бит – либо ноль, либо единица. Но квантовые компьютеры основаны на кубитах, а они могут находиться в состоянии суперпозиции, когда каждый кубит

– одновременно и ноль, и единица. И если для какого-нибудь расчета обычным компьютерам нужно, грубо говоря, выстроить последовательности, то квантовые вычисления происходят параллельно, в одно мгновение.

В компьютере Михаила Лукина таких кубитов – 51. Во-первых, он сделал систему, в которой больше всего кубитов. На всякий случай. На данный момент, я думаю, это больше чем в два раза больше кубитов, чем у кого-либо другого. И он специально сделал 51 кубит, а не 49, потому что *Google* все время говорил, что сделает 49», – объясняет гендиректор компании *Acronis*, сооснователь Российского квантового центра Сергей Белоусов.

Но для чего нам понадобятся квантовые компьютеры? Даже сами их создатели не знают наверняка. С их помощью могут быть разработаны совершенно новые материалы, сотни открытий в физике и химии. Квантовые компьютеры – пожалуй, единственное, что может приоткрыть тайну человеческого мозга и искусственного интеллекта.

Один из первых компьютеров был создан в 40-х годах XX века и весил 27 тонн. Если сравнить с современными устройствами, то обычный смартфон по мощности – это как 20 000 таких машин. И это за 70 лет прогресса. Но если наступит эра квантовых компьютеров, уже наши потомки будут удивляться, как вообще пользоваться этим антиквариатом.

2. Использование виртуальных машин

Понятие виртуальной машины

Впервые преподавание и обучение компьютерной науки началось в США в 40-е годы XX века в результате слияния теории алгоритмов и математической логики, а также изобретения электронных вычислительных машин, или иначе – компьютеров.

Появление компьютеров породило неизбежную тенденцию к появлению и последующему развитию информационного общества. А внедрение информационных технологий во все сферы деятельности человека способствовало возникновению и развитию глобального процесса информатизации.

Развитие компьютерной техники является необходимой составляющей процесса информатизации общества, в которой компьютер является универсальным техническим средством обработки информации любой сложности, который усиливает интеллектуальные возможности человека и общества.

Информатизация – это, в первую очередь, процесс, направленный на изменения в образе жизни людей. Данный процесс связан с использованием новых информационных и коммуникационных технологий в повседневной жизни для ликвидации компьютерной грамотности, и так далее.

Информатизация направлена на изменение технологий производства, на изменение материально-технической базы и подходов к ее построению, так как значительную роль приобретает использование аналитических и управляющих информационных систем, созданных на базе информационных и телекоммуникационных технологий.

Тенденция развития процесса информатизации повлекло к возникновению такого феномена, как информатизация общества.

Информатизация общества – это глобальный социальный процесс, особенность которого состоит в том, что преобладающим видом деятельности в сфере общественного производства является сбор, накопление, продуцирование, обработка, хранение, передача и использование информации, осуществляемые на основе современных средств микропроцессорной и вычислительной техники, а также на базе разнообразных средств информационного обмена.

Одним из приоритетных направлений процесса информатизации современного общества является информатизация образования. Информатизация образования является одним из важнейших условий реформирования и модернизации системы отечественного образования

Образовательная деятельность в настоящий момент тесно связана с процессом информатизации, а в силу быстрого темпа развития информационных технологий, их применение в образовательном процессе не всегда возможно своевременно осуществить.

В современном образовательном процессе, связанным с развитием информационных технологий, актуальным становится вопрос об использовании

различных видов информационных ресурсов для подготовки выпускников, конкурентоспособных на рынке труда.

Главными характеристиками выпускника любого образовательного учреждения являются его компетентность и профессиональная мобильность.

В условиях быстрого темпа развития информационных технологий использование морально устаревших операционных систем на новом современном компьютерном оборудовании становится нецелесообразным, а программное обеспечение может быть не совместимо с новейшими версиями операционных систем.

Так же, при изучении студентами некоторых дисциплин требуются большие материальные возможности, которыми порой не обладают учебные заведения.

К примеру, преподавание дисциплин, связанных с администрированием компьютерных сетей, подразумевает наличие мощной материальной базы. Многие учебные задачи, важные для становления молодого специалиста, трудно реализовать в классе с 10-15 компьютерами.

В связи с этим возникает потребность в использовании программных средств, призванных помочь организации учебного процесса.

Одним из способов решения данной проблемы является возможность использования виртуальных машин.

Виртуальная машина призвана «сгладить» процесс интеграции новых информационных технологий в образовательный процесс.

Использование виртуальных машин позволяет существенно расширить спектр учебных задач и улучшить качество подготовки выпускников, в частности специалистов в области информационных технологий.

Говоря о виртуальных машинах, следует разобраться, что представляет собой понятие виртуализация.

В широком смысле, понятие виртуализации представляет собой сокрытие настоящей реализации какого-либо процесса или объекта от истинного его представления для того, кто им пользуется. Продуктом виртуализации является нечто удобное для использования, на самом деле, имеющее более сложную или совсем иную структуру, отличную от той, которая воспринимается при работе с объектом. Иными словами, происходит отделение представления от реализации чего-либо. В компьютерных технологиях под термином «виртуализация» обычно понимается абстракция вычислительных ресурсов и предоставление пользователю системы, которая «инкапсулирует» (скрывает в себе) собственную реализацию. Проще говоря, пользователь работает с удобным для себя представлением объекта, и для него не имеет значения, как объект устроен в действительности.

Виртуализация представляет собой возможность запуска на физическом компьютере несколько изолированных друг от друга виртуальных машин, каждая из которых представляет собой полноценный компьютер, работающий на отдельном физическом компьютере.

Виртуальная машина имеет свой *BIOS,* оперативную память, жесткий диск (выделенные из реального компьютера) и способна эмулировать периферийные

устройства.

Прежде всего, виртуальная машина представляет собой папку с файлами; в зависимости от конкретной реализации их набор и количество могут меняться, но в основе любой виртуальной машины лежит один и тот же минимальный набор файлов. Это, прежде всего, файл с конфигурацией виртуальной машины и виртуальный жесткий диск.

Виртуальный жесткий диск виртуальной машины представляет собой файл, содержащий образ диска виртуальной машины. Он схож по структуре и содержанию с жестким диском реального компьютера. Виртуальный жесткий диск представляет наибольшую ценность, потому что его потеря равносильна отказу работы жесткого диска реального компьютера.

Следующим по важности файлом является конфигурация виртуальной машины, которая содержит описание эмулируемой аппаратной части виртуальной машины и выделенных ей ресурсов реального компьютера. К таким ресурсам можно отнести виртуальную оперативную память, которая является выделенной областью оперативной памяти реального компьютера.

По большому счету, потеря файла с конфигурацией виртуальной машины не критична. Если имеется только файл виртуального жесткого диска, то виртуальную машину можно запустить, предварительно создав вновь файл конфигурации. Аналогично с реальным компьютером жесткий диск можно подключить к другому компьютеру и получить вполне работоспособный компьютер.

В папке с виртуальной машиной могут быть и другие файлы, потеря которых не желательна, но и не критична.

Количество виртуальных машин на реальном компьютере ограничено размерами жесткого диска, а количество одновременно запущенных виртуальных машин ограничивается в основном количеством доступной оперативной памяти.

Взаимодействие виртуальных машин с реальным аппаратным обеспечением компьютера осуществляется через монитор виртуальных машин или гипервизор, через который происходит связь виртуальных машин с реальным компьютером.

Гипервизор (монитор виртуальных машин) – это платформа виртуализации, позволяющая запускать на одном физическом компьютере несколько операционных систем. Именно гипервизор предоставляет изолированное окружение для каждой виртуальной машины и дает операционной системе виртуальной машины доступ к аппаратному обеспечению реального компьютера.

С помощью гипервизора создаются виртуальные машины, для которых эмулируется (т. е. имитируется программными средствами работа какого- либо физического устройства) минимально необходимый набор виртуального «железа» и предоставляется доступ к отдельным ресурсам реального компьютера. Каждая виртуальная машина, как и обычный персональный компьютер, содержит собственный экземпляр операционной системы и прикладного программного

обеспечения, и последующее взаимодействие с ними ничем не отличается от работы с реальным компьютером.

Гипервизор может работать как в операционной системе реального компьютера, так и без нее, то есть гипервизор устанавливается поверх аппаратного обеспечения реального компьютера.

Операционную систему физического компьютера относительно операционной системы виртуальной машины принято называть основной или «хостовой» операционной системой, а систему, установленную на виртуальную машину, принято называть «гостевой» операционной системой.

На рис. 2 показано, как «гостевая» и «хостовая» операционные системы относятся друг к другу.

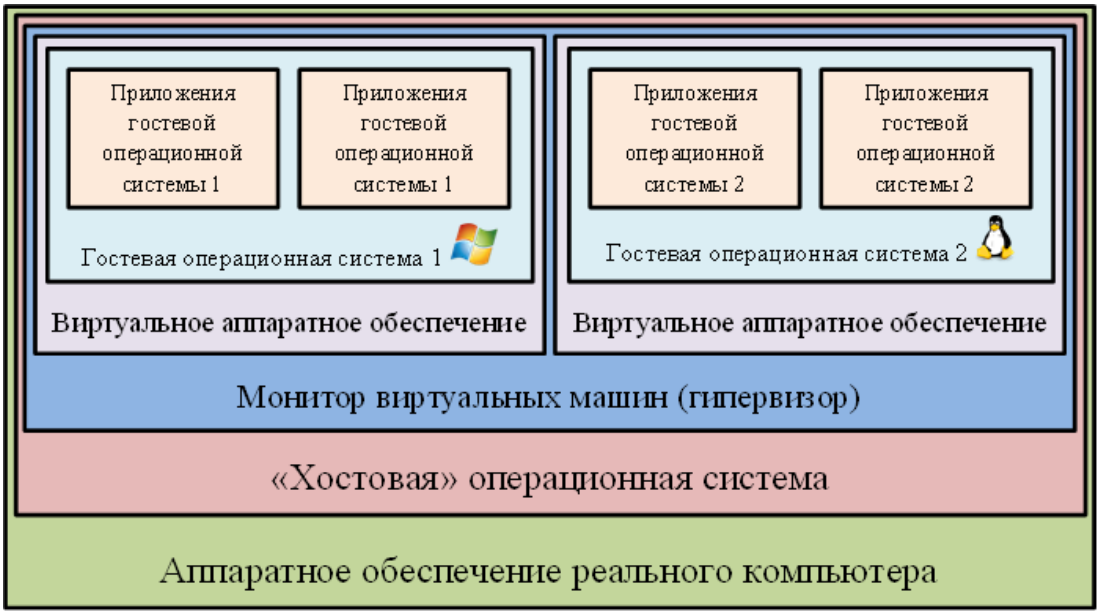


Рис. 2. Архитектура системы виртуальных машин

На самом деле виртуальная машина не имеет доступа к физическим ресурсам реального компьютера. «Хостовая» операционная система и монитор виртуальных машин разделяют между собой права на управление аппаратными компонентами компьютера, при этом «хостовая» операционная система занимается распределением ресурсов аппаратных компонентов между собственными приложениями, включая и монитор виртуальных машин. Гипервизор контролирует распределение ресурсов между запущенными виртуальными машинами, создавая для них иллюзию непосредственного доступа к аппаратным компонентам. «Гостевые» операционные системы в пределах выделенных им ресурсов управляют работой «своих» приложений.

«Гостевые» системы и «хостовая» операционная система работают одновременно, обмениваются данными и участвуют в сетевом взаимодействии не только с «хостовой» операционной системой, но и с внешней по отношению к физическому компьютеру сетью.

Виртуальная машина позволяет запускать отдельное приложение в своей собственной изолированной среде. Использование виртуальных машин решает проблему безопасности: приложение, запущенное в виртуальной машине, не способно нанести вред реальной операционной системе и другим приложениям. Таким образом, реальный компьютер огражден от возможных непреднамеренных действий пользователей.

Виртуальные машины позволяют запускать одновременно на одном реальном компьютере несколько различных операционных систем или конфликтующих друг с другом приложений.

На реальном компьютере может быть несколько виртуальных машин, каждая из которых имеет свою собственную аппаратную конфигурацию, например, количество процессоров, объем оперативной памяти и жесткого диска, наличие сетевых плат и других аппаратных компонентов. Эти ресурсы резервируются виртуальной машиной за счет физических ресурсов аппаратного обеспечения компьютера.

Возможности виртуальных машин достаточно широки. Перечислим лишь некоторые из них:

- возможность использования программ, которые не поддерживаются «хостовой» операционной системой реального компьютера;
- защищенность информации на реальном компьютере, так как виртуальная машина работает изолированно от реального компьютера – всевозможные вирусы и вредоносное программное обеспечение сможет лишь повредить «гостевую» операционную систему виртуальной машины, не затронув реальную систему;
- возможность экспериментирования с системой. Например, изменение параметров реестра с целью их изучения. Любые изменения в «гостевой» операционной системе виртуальной машине не нанесет вреда «хостовой» операционной системе реального компьютера;
- большие возможности обучения работе с различными операционными системами и программами. Например, можно создать несколько виртуальных машин с различными операционными системами, и учиться работе с ними;
- эмуляция компьютерной сети с помощью нескольких виртуальных машин;
- простота создания резервной копии операционной системы. Не придется создавать образы диска, всего лишь требуется скопировать папку с файлами виртуальной машины;

Виртуальные машины также имеют и свои недостатки:

- потребность в наличии достаточных аппаратных ресурсов для функционирования нескольких операционных систем одновременно;
- операционная система работает несколько медленнее в виртуальной машине, нежели на аппаратном обеспечении реального компьютера.

Недостатки виртуальных машин являются в принципе разрешимыми и, по сравнению с возможностями, являются не столь существенными.

На сегодняшний день вычислительные мощности персональных

компьютеров достигли такого уровня, когда один физический компьютер может поддерживать несколько одновременно запущенных операционных систем в виртуальных машинах. До недавнего времени виртуальные машины были чем-то необычным для конечных пользователей, которые устанавливали их в основном в ознакомительных целях.

Понятие виртуализации

Понятие «виртуализация» – одно из наиболее распространенных терминов информационных технологиях, однако строгого определения этого понятия практически нет. Произведя анализ предлагаемых определений данного термина можно сформулировать следующее определение виртуализации: это набор средств и технологий, позволяющих распределять совокупность ресурсов вычислительной системы между многими средами выполнения приложений.

В настоящее время исследователи в области информационных технологий сходятся в классификации понятия виртуализации, а именно его условного подразделения на две фундаментально различающиеся категории, а именно: виртуализации платформ и виртуализации ресурсов.

Под *виртуализацией платформ* следует понимать создание программных систем на основе существующих аппаратно-программных комплексов, зависящих или независящих от них. Система, предоставляющая аппаратные ресурсы и программное обеспечение, называется «хостовой», а симулируемые ей системы – «гостевыми». Чтобы «гостевые» системы могли стабильно функционировать на платформе «хостовой» системы, необходимо, чтобы программное и аппаратное обеспечение хоста было достаточно надежным и предоставляло необходимый набор интерфейсов для доступа к его ресурсам. Есть несколько видов виртуализации платформ, в каждом из которых осуществляется свой подход к понятию «виртуализация». Виды виртуализации платформ зависят от того, насколько полно осуществляется симуляция аппаратного обеспечения.

До сих пор нет единого мнения об устоявшихся понятиях и дефинициях в сфере виртуализации, поэтому следующее деление на подвиды – следствие анализа многих источников вопросов виртуализации.

Хотя данная классификация весьма условна, виртуализацию платформ в основном подразделяют на несколько видов:

1. **Полная эмуляция (симуляция).** При таком виде виртуализации виртуальная машина полностью виртуализирует все аппаратное обеспечение при сохранении «гостевой» операционной системы в неизменном виде. Такой подход позволяет эмулировать различные аппаратные архитектуры. Например, можно запускать виртуальные машины с «гостевыми» системами для 32-разрядных процессоров на платформах с другой архитектурой.

Долгое время такой вид виртуализации использовался, чтобы разрабатывать программное обеспечение для новых процессоров еще до того, как они были физически доступными. Такие эмуляторы также применяют для низкоуровневой

отладки операционных систем. Основной минус данного подхода заключается в том, что эмулируемое аппаратное обеспечение существенно замедляет быстродействие «гостевой» системы, что делает работу с ней очень неудобной, поэтому, кроме как для разработки системного программного обеспечения, а также образовательных целей, такой подход малоэффективен.

1. **Частичная эмуляция (нативная виртуализация, *native virtual- ization).*** В этом случае виртуальная машина виртуализирует лишь необходимое количество аппаратного обеспечения, чтобы она могла быть запущена изолированно. Такой подход позволяет запускать «гостевые» операционные системы, разработанные только для той же архитектуры, что и у хоста. Таким образом, несколько экземпляров «гостевых» систем могут быть запущены одновременно. Этот вид виртуализации позволяет существенно увеличить быстродействие «гостевых» систем по сравнению с полной эмуляцией и широко используется в настоящее время. Кроме того, в целях повышения быстродействия в платформах виртуализации, использующих данный подход, применяется специализированное промежуточное звено между «гостевой» операционной системой и оборудованием (гипервизор), позволяющая «гостевой» системе напрямую обращаться к ресурсам аппаратного обеспечения. Гипервизор, именуемый также монитором виртуальных машин – одно из ключевых понятий в мире виртуализации. Применение гипервизора, являющегося связующим звеном между «гостевыми» системами и аппаратурой, существенно увеличивает быстродействие платформы, приближая его к быстродействию физической платформы.

К достоинствам данного подхода можно причислить относительную простоту реализации, универсальность и надежность решения; все функции управления берет на себя «хостовая» операционная система.

К минусам данного вида виртуализации можно отнести зависимость виртуальных машин от архитектуры аппаратной платформы.

2. **Частичная виртуализация, а также «виртуализация адресного пространства» («*address space virtualization*»).** При таком подходе, виртуальная машина симулирует несколько экземпляров аппаратного окружения (но не всего), в частности, пространства адресов. Такой вид виртуализации позволяет совместно использовать ресурсы и изолировать процессы, но не позволяет разделять экземпляры «гостевых» операционных систем. Строго говоря, при таком виде виртуализации пользователем не создаются виртуальные машины, а происходит изоляция каких-либо процессов на уровне операционной системы. В данный момент многие из известных операционных систем используют такой подход.

3. **Паравиртуализация (*paravirtualization*).** При применении паравиртуализации нет необходимости симулировать аппаратное обеспечение, однако, вместо этого (или в дополнение к этому), используется специальный программный интерфейс (*API, application programming interface*) для взаимодействия с «гостевой» операционной системой. Это требует поддержки со стороны производителей операционных систем, которые слабо верят в

возможности такого метода виртуализации, в связи с чем этот вид виртуализации развивается очень слабо, хотя и существуют сравнения, показывающие, что быстродействие паравиртуализации выше.

4. **Виртуализация уровня операционной системы (виртуализация систем).** Данный вид виртуализации применяется в целях создания нескольких защищенных виртуальных серверов на одном физическом.

«Гостевая» система, в данном случае, разделяет использование одного ядра операционной системы хостинга с другими «гостевыми» операционными системами. Виртуальная машина представляет собой окружение для приложений, запускаемых изолированно. Данный тип виртуализации применяется при организации систем хостинга, когда в рамках одного экземпляра ядра требуется поддерживать несколько виртуальных серверов клиентов.

5. **Виртуализация уровня приложений.** Этот вид виртуализации не похож на все остальные: если в предыдущих случаях создаются виртуальные среды или виртуальные машины, использующиеся для изоляции приложений, то в данном случае само приложение помещается в контейнер с необходимыми элементами для своей работы: файлами реестра, конфигурационными файлами, пользовательскими и системными объектами. В результате получается приложение, не требующее установки на аналогичной платформе. При переносе такого приложения на другую машину и его запуске, виртуальное окружение, созданное для программы, разрешает конфликты между ней и операционной системой, а также другими приложениями. Такой способ виртуализации похож на поведение интерпретаторов различных языков.

В рамках получения преимуществ на основе применения технологии виртуализации для образовательного процесса особый интерес представляют виртуализация систем и виртуализация приложений.

Если виртуализация платформ преимущественно относится к процессу создания виртуальных машин, то виртуализация ресурсов обобщает в себе подходы к созданию виртуальных систем. Виртуализация ресурсов позволяет концентрировать, абстрагировать и упрощать управление группами ресурсов, таких как сети, хранилища данных и пространства имен.

Виртуализация ресурсов, в отличие от виртуализации платформ, имеет более широкий и расплывчатый смысл и представляет собой массу различных подходов, направленных на повышение удобства обращения пользователей с системами в целом. Поскольку виртуализация платформ ориентирована больше на конкретные цели, то ее применение может повысить эффективность использования компьютерного оборудования.

Сфера применения виртуализации достаточна широка. На сегодняшний день можно обозначить следующие варианты использования продуктов виртуализации:

1. **Консолидация серверов.** В данный момент приложения, работающие на серверах в *IT*-инфраструктуре (*IT – Information Technology*, информационные технологии) компаний, либо иных учреждениях, создают небольшую нагрузку

на аппаратные ресурсы серверов. Виртуализация позволяет разместить все на одном физическом сервере, увеличив его загрузку, повысив тем самым эффективность использования аппаратуры, что позволяет существенно сэкономить на оборудовании, обслуживании и электроэнергии.

2. **Разработка и тестирование приложений.** Множество продуктов виртуализации позволяют запускать несколько различных операционных систем одновременно, позволяя тем самым разработчикам и тестерам программного обеспечения тестировать их приложения на различных платформах и конфигурациях. Также удобные средства по созданию «снимков» текущего состояния системы одним кликом мыши и такого же простого восстановления из этого состояния, позволяют создавать тестовые окружения для различных конфигураций, что существенно повышает скорость и качество разработки.

3. **Использование в бизнесе.** Этот вариант использования виртуальных машин является наиболее обширным и творческим. К нему относится все, что может понадобиться при повседневном обращении с *IT*- ресурсами в бизнесе. Например, на основе виртуальных машин можно легко создавать резервные копии рабочих станций и серверов (просто скопировав папку), строить системы, обеспечивающие минимальное время восстановления после сбоев и т. п. К данной группе вариантов использования относятся все те бизнес-решения, которые используют основные преимущества виртуальных машин.

4. **Использование виртуальных рабочих столов.** *VDI (Virtual Desk- top Infrastructure*) – это технология, позволяющая создавать виртуальную *IT*-инфраструктуру и разворачивать полноценные рабочие места на базе одного сервера, на котором работает множество виртуальных машин.

Виртуальные персональные компьютеры, с которыми работают пользователи, ничем не отличаются от обычных персональных компьютеров. Однако технология *VDI* позволяет держать всю необходимую для работы информацию под рукой в любом месте, где есть доступ к Интернету (дома, в деловых поездках, на отдыхе). Удалённая работа становится очень простой и комфортной. Вся информация из виртуального компьютера хранится в специальных дата-центрах. Это обеспечивает её гарантированную защиту от потери (все данные проходят процедуру автоматического резервного копирования), случайного или преднамеренного удаления, а также от доступа к ней посторонних лиц.

Обслуживание виртуальных рабочих мест (в частности установка программного обеспечения, обновление приложений и т.д.) производится централизовано, что позволяет минимизировать временные затраты, а также значительно снизить нагрузку на системных администраторов. Также происходит экономия денежных средств на покупке лицензионного программного обеспечения для каждого компьютера и содержании дополнительного штата специалистов.

Технология *VDI* позволяет развернуть свое рабочее место и получить доступ ко всей информации не только с персонального компьютера, но и ноутбука, смартфона или другого авторизованного устройства.

Все перечисленные варианты использования виртуальных машин фактически являются лишь сферами их применения на данный момент.

На сегодняшний день проекты по виртуализации *IT*-инфраструктуры активно внедряются многими ведущими компаниями, занимающимися системной интеграцией и являющимися авторизованными партнерами провайдеров систем виртуализации. В процессе виртуализации *IT*- инфраструктуры создается виртуальная инфраструктура – комплекс систем на основе виртуальных машин, обеспечивающих функционирование всей IT- инфраструктуры, обладающий многими новыми возможностями при сохранении существующей схемы деятельности IT-ресурсов. Разработчики различных платформ виртуализации могут предоставить информацию об успешных проектах по внедрению виртуальной инфраструктуры в крупных банках, промышленных компаниях, больницах, образовательных учреждениях.

Идея, заложенная в технологиях виртуализации, открывает широкие возможности по их использованию. Потому что, в конечном счете, все делается для удобства пользователя и упрощения использования привычных ему вещей.

Место и роль виртуальных машин в образовательном процессе

Сферы пользования виртуальных машин на сегодняшний день достаточно широки, как и сами возможности виртуальных машин. Использовать их можно как у себя дома, так и в других учреждениях, предварительно скопировав ее на любой современный съемный носитель информации, при условии, что на физическом компьютере будет установлена та же платформа виртуализации, использующаяся вашей виртуальной машиной.

В современном образовательном процессе, связанном с использованием информационных технологий, а также в условиях быстрого их развития некоторые использующиеся программы могут быть несовместимы с современными версиями операционных систем. В связи с этим возникает потребность в использовании виртуальных машин.

Использование виртуальных машин в учебном процессе может помочь операционной системе реального компьютера прослужить более длительное время, избавив память компьютера от заполнения ее ненужными файлами.

Использование виртуальных машин позволяет существенно расширить спектр учебных задач и улучшить качество подготовки выпускников, в частности специалистов в области информационных технологий.

Использование компьютеров в образовании на сегодняшний день является неотъемлемой частью образовательного процесса. Применяются они не только для подготовки специалистов информационных и коммуникационных технологий, но и многих других направлений подготовки, профилей и профилизаций. К примеру, для подготовки специалистов к работе на различных станках, где имитируется автоматизация каких-либо процессов, имитирование различных учебных ситуаций: учебные тренажеры, симуляторы, программы для

работы с техникой и т.д. Вся эта среда обусловлена наличием персонального компьютера.

В силу развития информационных технологий, на современных компьютерах используются новейшие версии операционных систем. Использование старых версий становится нецелесообразным.

Но не всегда новейшие версии операционных систем совместимы с интересующими нас приложениями. Проблема совместимости решается использованием виртуальных машин.

Виртуальные машины могут использоваться в учебном процессе при изучении соответствующ*их* дисциплин. Например, в *Томском исследовательском политехническом университете в инженерной школе информационных технологий и робототехники* преподаются дисциплины по таким направлениям, как компьютерная графика, программирование, операционные системы, компьютерные сети*, администрирование вычислительных систем и сетей* и т. д. Существует огромный перечень необходимого учебного программного обеспечения. Виртуальные машины являются файлами, их копирование позволяет избежать непосредственной установки учебного программного обеспечения на каждый компьютер.

Для этого на виртуальную машину устанавливаются операционная система и определенные приложения. Студенты на практическом занятии при изучении той или иной дисциплины запускают соответствующую виртуальную машину с установленными на ней приложениями, после чего работают уже непосредственно в виртуальной машине. Все созданные при работе файлы сохраняются уже на виртуальной машине, что избавляет реальный компьютер от их непосредственного хранения.

В случае краха операционной системы и дальнейшей ее переустановки на реальном компьютере уже не возникает потребность в повторной установке приложений, поскольку виртуальную машину можно будет снова запустить. Для этого достаточно будет установить тот же гипервизор, при условии, что виртуальный жесткий диск не будет утерян и скопировать ранее созданную виртуальную машину.

Благодаря использованию виртуальных машин в образовательном процессе снимается масса проблем:

- снижается риск от неверных действий обучаемых;
- легкость восстановления, в случае краха виртуальной машины;
- простота конфигурирования и использования виртуальных машин;
- решается проблема совместимости приложений с операционной системой.

Виртуальные машины решают проблему нанесения вреда на реальном компьютере, так как она изолирована от реальной операционной системы. А в случае выхода из строя самой виртуальной машины, восстановление ее не составляет большого труда. Оно осуществляется путем простого копирования нескольких файлов оригинала виртуальной машины. Это значительно быстрее того, если бы мы каждый раз переустанавливали операционную систему на

реальном компьютере. Тем самым, использование виртуальных машин в учебном процессе является безопасным шагом и рациональным решением.

Обзор наиболее известных виртуальных машин

Виртуальных машин, или иначе платформ виртуализации, имеется множество. Дадим объективную оценку, а также проведем сравнительный анализ, на предмет возможного применения виртуальных машин в образовательном процессе.

Рассмотрим решения для локального использования виртуальных машин.

VMware. Начнем с самой известной, платформы виртуализации от американской компании, крупнейшего разработчика программного обеспечения для виртуализации «*VMware*».

VMware, Inc. была основана в 1998 году. Ее основатели: Дайана Грин (*Diane Greene*), Мендель Розенблюм (*Mendel Rosenblum*), Скот Девайн (*Scott Devine),* Эдвард Ванг (*Edward Wang*) и Эдуард Багнион (*Edouard Bugnion*). Последний оставался в компании до 2005 года, после чего основал собственную *Nuova Systems* (ныне принадлежит *Cisco*).

Свой первый продукт (*VMware Workstation*) компания представила в 1999 году. В 2001 году появляются первые серверные приложения. В 2003 году, компания становится безусловным лидером с миллиардными доходами. Продукция компании неоднократно удостаивалась престижных наград. В 2004 году *VMware, Inc.* приобрела *EMC Corporation.* В ноябре 2008 года *VMWare* поглощает компанию *Tungsten Graphics,* с целью улучшения поддержки 3D-графики в продуктах виртуализации.

Логотипом *VMware, Inc.* служат несколько перекрывающих друг друга квадратов, символизирующих, по все видимости, окна с запущенными операционными системами. Название произошло от аббревиатуры «*VM*», означающей «*Virtual Machine*», скомбинированной с «*ware*» – второй частью от слова «*software*» (рис. 3).

Рис.3. Логотип компании *VMware*

Штаб-квартира компании находится в Пало Альто, Калифорния, США (*Palo Alto, California, USA*).

На сегодняшний день компания *VMware* является флагманом в производстве систем виртуализации. Её продукты охватывают практически все сферы использования виртуальных машин – от программного обеспечения для домашних персональных компьютеров, до мощных высокопроизводительных серверных решений. Необходимо отметить, что отсутствуют возможности одновременного использования каких-либо двух продуктов *VMware* одновременно на одном физическом хосте (за исключением *VMware Virtual*

Center), а также установки любого из продуктов внутри виртуальной машины.

Рассмотрим подробнее программные решения от *VMware*.

VMware Workstation – система виртуализации для домашних персональных компьютеров, предназначенная для использования разработчиками или тестировщиками и другими специалистами в области информационных технологий, в случаях, когда необходим запуск нескольких операционных систем на одном компьютере.

Это идеальное решение в случае, когда необходимо проверить какой- либо программный продукт на работоспособность в различных операционных системах и их конфигурациях. Огромным преимуществом данной системы является возможность создания мгновенных снимков, или «снапшотов» (*snapshot*, снимок файловой системы), операционной системы и дисков, которые впоследствии могут быть использованы для отката состояния операционной системы и содержимого дисков. Также этот продукт поддерживает создание деревьев «снапшотов», что невероятно полезно при конфигурировании системы с различными надстройками (к примеру, различными версиями браузера *Internet Explorer)*.

VMware Player – как видно из его названия это «проигрыватель» виртуальных машин. Предназначен для использования виртуальных машин, заранее подготовленных в какой-либо из систем виртуализации *VMware* (например, *Workstation*). В нём может быть запущено не более одной виртуальной машины (но таких проигрывателей может быть запущено несколько). Также есть возможность создания «снапшотов». Он бесплатен, что является большим плюсом.

Раньше, бесплатная программа виртуализации *VMware Player* умела только запускать созданные ранее виртуальные машины (их образы), но сейчас она научилась еще и создавать их. То есть приложение стало полноценным инструментом виртуализации. Правда в отличии от платного своего аналога, именуемого как *VMware Workstation*, функционал приложения несколько ограничен, но можно смело сказать, что расширенные функции платной версии далеко не всегда нужны домашнему пользователю. В качестве основной операционной системы *VMware Player* использует *Linux* и *Windows*, а в качестве «гостевых» операционных систем, могут быть использованы *DOS*, *Windows*, *Linux*, *Mac*, *BSD* и др.

Также выпущено приложение *VMware Fusion*, которое по своим возможностям полностью напоминает *VMware Workstation*. Одной из отличительных особенностей данного приложения является режим *Unity*, который предназначен для скрытия окна виртуальной машины с глаз пользователя. То есть пользователь видит, как бы операционную систему в операционной системе. При этом окна, открытых в «гостевой» операционной системе приложений, отображаются на основном рабочем столе.

Oracle. Американская корпорация, второй по величине доходов производитель программного обеспечения, крупнейший производитель программного обеспечения для организаций, крупный поставщик серверного

оборудования.

Одной из самых распространенных программ для виртуализации является *VirtualBox*. Над созданием этого приложения трудилась не одна группа разработчиков, и далеко не одна именитая в *IT*-среде компания (рис. 4).

Рис. 4. Логотип виртуальной машины *VirtualBox* от компании *Oracle*

В настоящее время продуктом полностью владеет компания *Oracle*, которая получила ее в процессе поглощения предыдущего владельца (*Sun Microsystems*) еще в 2010 году. Результате всех перемещений программы *Vir- tualBox* от одного владельца к другому, на сегодняшний день мы имеем программу виртуализации с большим количеством поддерживаемых операционных систем. На официальном сайте программы, можно скачать для установки на компьютер сборки для платформ *Windows, Mac OS, Linux* и *Solaris*. Эти же операционные системы кстати говоря можно использовать в *VirtualBox* в качестве «гостевых». В целом *VirtualBox* распространяется с открытым исходным кодом, что делает ее полностью бесплатной для использования. Однако, для получения более расширенного функционала и возможностей, скажем для поддержки протокола *RDP (Remote Desktop Proto- col*, протокол удаленного рабочего стола) или стандарта *USB* 3.0 для «гостевой» операционной системы, пользователю потребуется устанавливать дополнительные плагины, которые к слову тоже распространяются бесплатно, но имеют закрытый исходный код.

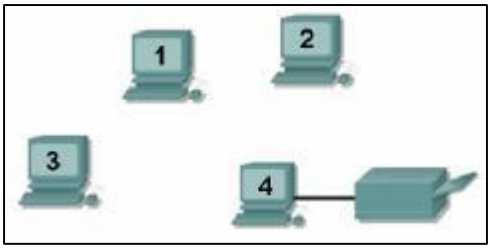
Microsoft. Компания *Microsoft* – разработчик самой популярной операционной системы, разработала собственную систему виртуализации, которая предназначена для работы исключительно в среде *Windows* – это *Virtual PC* 2007 для *Windows XP* и *Vista*, а также сравнительно новая виртуальная машина *Virtual PC* для Windows *7*. В дополнение к последней версии программы виртуализации разработчики создали инструмент *Win- dows XP Mode*, который представляет собой виртуальную версию *XP Profes- sional*. Данный инструмент позволяет запускать устаревшие приложения и программы в более новой операционной системе *Windows* 7 (рис. 5).

Рис. 5. Логотип виртуальной машины *Virtual PC* от компании *Microsoft*

В *Windows* 8 и тестовой *Windows* 10 на смену *Virtual PC* пришла более новая технология *Hyper-V,* позаимствованная из *Windows Server*. Данная технология имеет массу отличий от обычных виртуальных машин, так как больше ориентирована на серверную архитектуру.

3. **Введение в сетевые технологии**

Первые компьютеры были автономными устройствами. Каждый компьютер работал отдельно, независимо от других. При таком подходе возникало много проблем. Например, есть сеть, в которой к одному компьютеру подключен



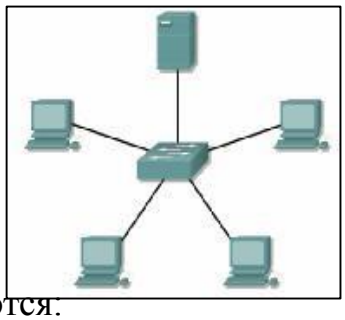
принтер. В этом случае, использовать принтер мог человек, работавший за этим компьютером, другие сотрудники не имели возможности распечатывать свои документы. Так же возникали трудности при работе над одним документом нескольких сотрудниками. При изменении файла требовалось каждый раз производит обновление у всех остальных сотрудников. При таком подходе была очень низкая эффективность работы. Необходимо было найти решение, которое бы удовлетворяло трем перечисленным ниже требованиям, а именно:

- Устраняло дублирование оборудования и ресурсов.
- Обеспечивало эффективный обмен данными между устройствами.
- Снимало .проблему управления сетью.

Было найдено два решения, выполняющих поставленные условия. И это были локальные и глобальные сети.

Локальные сети

Локальные вычислительные сети (ЛВС) – это высокоскоростные сети с малым количеством ошибок, которые охватывают небольшие географические пространства (до нескольких тысяч метров). ЛВС объединяют рабочие станции, терминалы и периферийные устройства в одном здании или другой пространственно ограниченной области. Локальные сети обеспечивают множеству подключенных настольных устройств доступ к среде передачи данных с высокой пропускной способностью



Характерными особенностями локальной сети являются:

- ограниченные географические пределы;
- обеспечение многим пользователям доступа к среде с высокой пропускной способностью;
- постоянное подключение к локальным сервисам;
- физическое соединение рядом стоящих устройств.

К устройствам локальной сети относятся следующие устройства:

- *Мосты* – подключают сегменты локальной сети и помогают фильтровать трафик.

- *Концентраторы* – концентрируют соединения локальной сети и позволяют использовать в качестве среды передачи данных витую пару.
- *Коммутаторы Ethernet* – обеспечивают сегментам и настольным системам полнодуплексную связь и выделенную полосу пропускания.
- *Маршрутизаторы* – обеспечивают большое количество сервисов, включая организацию взаимодействия сетей и управление широковещанием.

Наиболее распространенными технологиями ЛВС являются;
- *Ethernet,*
- *Fiber Distributed Data Interface (FDDI)*
- *Token Ring*

Глобальные сети

Быстрое распространение компьютеров привело к увеличению числа локальных сетей. Каждая локальная сеть представляла отдельный электронный «остров», не имеющий связи с другими локальными сетями. Требовалось найти способ передачи информации от одной локальной сети к другой. Решить эту задачу помогло создание глобальных сетей. Глобальные сети служат для объединения локальных сетей и обеспечивают связь между компьютерами, находящимися в локальных сетях. Глобальные сети охватывают значительные географические пространства и дают возможность связать устройства, расположенные на большом удалении друг от друга.

При подключении компьютеров, принтеров и других устройств к глобальной сети возникает возможность совместного использования информации и ресурсов, а также доступа к *Internet*.

Распределенные сети состоят из трех основных компонент:
- Локальные сети, как узлы распределенной сети.
- Каналы, соединяющие ЛВС.
- Оборудование и программы, обеспечивающие локальным сетям доступ к каналам связи.

Для объединения локальных сетей требуется специальное оборудование независимо от того, находятся ли эти ЛВС в одном здании или связаны через распределенную сеть.

- *Повторители (Repeater)* – усиливают полученный из кабельного сегмента сигнал и передают его в другой сегмент.

Объединяют идентичные ЛВС.

Простое усиление сигналов.

- *Мосты (Bridge)* передают сообщения на основе записей в таблице пересылки.

Возможность фильтрации сетевого трафика.

Сохраняет информацию обо всех узлах.

Соединяет идентичные или разные сети (например, Ethernet и

Token Ring).

- *Маршрутизаторы (Router)* обеспечивают выбор маршрута обмена данными между узлами сети.

Принимает решение о выборе "лучшего пути".

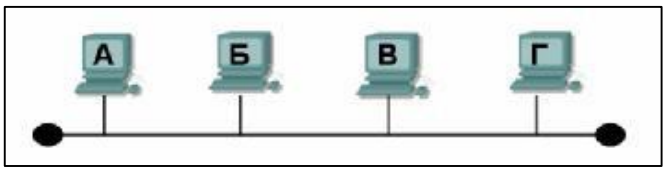
Дистанция обычно оценивается в интервалах (*hop*) – промежутках между двумя соседними маршрутизаторами на пути от отправителя к получателю.

Архитектура сети

Сетевая архитектура сродни архитектуре строений. Архитектура здания отражает стиль конструкций и материалы, используемые для постройки. Архитектура сети описывает не только физическое расположение сетевых устройств, но и тип используемых адаптеров и кабелей. Кроме того, сетевая архитектура определяет методы передачи данных по кабелю.

Топология сети описывает схему физического соединения компьютеров. Существуют 3 основных типа сетевой топологии:

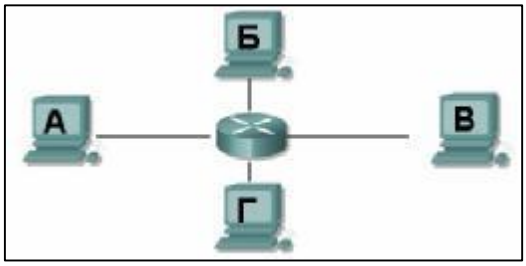
Общая шина



При использовании шинной топологии компьютеры соединяются в одну линию, по концам, которой устанавливают терминаторы. Когда источник передает сигналы в сетевую среду, они движутся в обоих направлениях от источника. Эти сигналы доступны всем устройствам в ЛВС. Каждое устройство проверяет проходящие данные. Если *MAC*- или *IP*-адрес пункта назначения, содержащийся в пакете данных, не совпадает с соответствующим адресом этого устройства, данные игнорируются.

Преимущества шинной топологии заключаются в простоте организации сети и низкой стоимости. Недостатком является низкая устойчивость к повреждениям – при любом обрыве кабеля вся сеть перестает работать, а поиск повреждения весьма затруднителен.

Звезда



При использовании топологии "звезда", каждый компьютер подключается к специальному концентратору (хабу). Связь между устройством и центральным каналом или концентратором осуществляется посредством двухточечных линий.

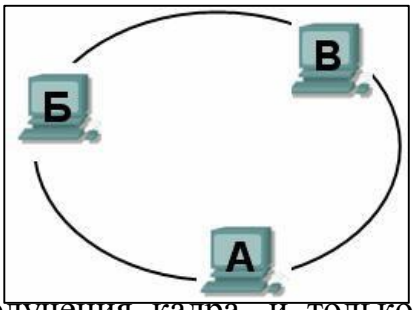
Когда источник передает сигналы в сетевую среду, данные посылаются центральному сетевому устройству (концентратору), затем концентратор переправляет их устройству в соответствии с адресом, содержащимся в данных.

Преимущества топологии звезда:

- простота обслуживания – единственной областью концентрации является центр сети;
- топология позволяет легко диагностировать проблемы и изменять схему прокладки;
- к сети, использующей звездообразная топология легко добавлять рабочие станции;
- если выходит из строя один из участков, то теряет связь только устройство, подключенное к этой точке, остальная часть сети будет функционировать нормально;
- звездообразная топология считается надежной.

Главным недостатком такой топологии является выход из строя центрального сетевого устройства, в этом случае сеть становится не работоспособной.

Кольцо



При такой топологии узлы сети образуют виртуальное кольцо (концы кабеля соединены друг с другом). Каждый узел сети соединен с двумя соседними. Кольцевая топология - кадр управления (*supervisory frame*) называемый также маркером (*token*) последовательно передается от станции к соседней. Станция, которая хочет получить доступ к среде передачи, должна ждать получения кадра, и только после этого может начать передачу данных. Преимуществом кольцевой топологии является ее высокая надежность (за счет избыточности), однако стоимость такой сети достаточно высока за счет расходов на адаптеры, кабели и дополнительные приспособления.

Сетевой уровень и маршрутизация

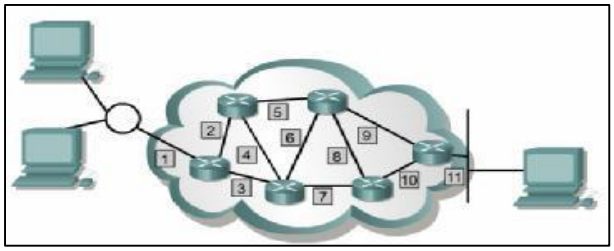
Каким путем должен пойти трафик через сети. Этот выбор пути происходит на сетевом уровне. Функция выбора пути позволяет маршрутизатору оценивать имеющиеся пути до пункта назначения и устанавливать наилучший в этом плане метод обработки пакетов.

Оценивая возможные пути, протоколы маршрутизации используют информацию о топологии сетей. Эта информация может конфигурироваться сетевым администратором или собираться посредством динамических процессов, исполняемых в сети.

Сетевой уровень для сетей играет роль интерфейсов и обеспечивает своему пользователю, транспортному уровню, сервис по наилучшей сквозной доставке пакетов. Сетевой уровень пересылает пакеты из сети-источника в сеть пункта назначения на основе таблицы *IP*-маршрутизации.

После того как маршрутизатор определит, какой путь использовать, он может переходить к коммутированию пакета: принимая пакет, полученный через один интерфейс, и перенаправляя его на другой интерфейс или порт, который соответствует наилучшему пути к пункту назначения пакета.

Чтобы иметь практическую ценность, сеть должна непротиворечивым образом показывать пути, имеющиеся между маршрутизаторами. Как показано на рисунке, каждая связь между маршрутизаторами имеет номер, который маршрутизаторы используют в качестве адреса. Эти адреса должны нести в себе информацию, которая может быть использована в процессе маршрутизации. Это означает, что адрес должен содержать информацию о пути соединений сред передачи данных, которую процесс маршрутизации будет использовать для пересылки пакетов от отправителя в конечный пункт назначения.



Используя эти адреса, сетевой уровень может обеспечить организацию релейного соединения, которое будет связывать независимые сети. Непротиворечивость адресов уровня 3 во всем многосетевом комплексе также улучшает использование полосы пропускания, исключая необходимость в широковещательных рассылках. Широковещательные рассылки приводят к накладным расходам в виде ненужных процессов и напрасно расходуют мощности устройств или каналов связи, которым вовсе не надо принимать эти широковещательные рассылки.

Благодаря использованию непротиворечивой сквозной адресации для представления пути соединений сред передачи данных сетевой уровень может находить путь до пункта назначения без ненужной загрузки устройств или каналов связи многосетевого комплекса широковещательными рассылками.

Адресация: сеть и хост-машина

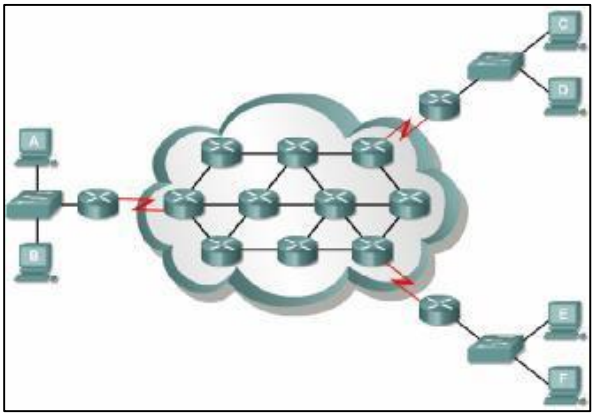
Сетевой адрес состоит из сетевой части и части хост-машины, которые используются маршрутизатором в "облаке" сети. Обе они нужны для доставки пакетов от отправителя получателю. Сетевой адрес используется маршрутизатором для идентификации сети отправителя или получателя пакета внутри сетевого комплекса.

Для некоторых протоколов сетевого уровня эти отношения задаются администратором сети, который назначает сетевые адреса в соответствии с планом адресации сетевого комплекса. Для других же протоколов сетевого

уровня назначение адресов является частично или полностью динамическим. Большинство схем адресации в сетевых протоколах использует некоторую форму адреса хост-машины или узла.

Маршрутизация с использованием сетевых адресов

В общем случае маршрутизаторы передают пакет по эстафете из одного канала связи: другой. Чтобы осуществить такую эстафетную передачу, маршрутизаторы используют, основные функции: функцию определения пути и функцию коммутации.



На рисунке показано, как маршрутизаторы используют адресацию для реализации своих функций маршрутизации и коммутации. Сетевая часть адреса используется для осуществления выбора пути, а узловая часть адреса говорит о порте маршрутизатора" по пути следования.

Маршрутизатор отвечает за передачу пакета в следующую сеть по пути следования. Сетевая часть адреса используется маршрутизатором для выбора пути. Функция коммутирования позволяет маршрутизатору принимать пакет на один интерфейс и переправлять его на другой. Функция определения пути позволяет маршрутизатору выбрать наиболее подходящий интерфейс для переадресации пакета. Узловая часть адреса говорит о конкретном порте маршрутизатора, который имеет выход на соседний маршрутизатор в выбранном направлении.

Протоколы маршрутизации и маршрутизируемые протоколы

Маршрутизируемый протокол – любой сетевой протокол, который обеспечивает в адресе сетевого уровня достаточно информации, чтобы позволить передать пакет от одной хост-машины к другой на основе принятой схемы адресации. Маршрутизируемый протокол определяет формат и назначение полей внутри пакета. В общем случае пакеты переносятся от одной конечной системы к другой. Примером маршрутизированного протокола является межсетевой протокол *IP*.

Протокол маршрутизации – поддерживает маршрутизируемый протокол за счет предоставления механизмов коллективного использования маршрутной информации. Сообщения протокола маршрутизации циркулируют между маршрутизаторами. Протокол маршрутизации позволяет маршрутизаторам обмениваться информацией с другими маршрутизаторами с целью актуализации и ведения таблиц. Примерами протоколов маршрутизации являются протокол

маршрутной информации (*RIP*), протокол внутренней маршрутизации между шлюзами (*IGRP*), усовершенствованный протокол внутренней маршрутизации между шлюзами (*EIGRP*) и протокол маршрутизации с выбором кратчайшего пути (*OSPF*).

Статические и динамические маршруты

Статическая информация администрируется вручную. Сетевой администратор вводит ее в конфигурацию маршрутизатора. Если изменение в топологии сети требует актуализации статической информации, то администратор сети должен вручную обновить соответствующую запись о статическом маршруте.

Динамическая информация работает по-другому. После ввода администратором сети команд, запускающих функцию динамической маршрутизации, сведения о маршрутах обновляются процессом маршрутизации автоматически сразу после поступления из сети новой информации. Изменения в динамически получаемой информации распространяются между маршрутизаторами как часть процесса актуализации данных.

Статическая маршрутизация имеет несколько полезных применений, которые связаны с привлечением специальных знаний администратора сети о сетевой топологии. Одним из таких применений является защита в сети. Динамическая маршрутизация раскрывает все, что известно о сети. Однако по причинам безопасности может понадобиться скрыть некоторые части сети. Статическая маршрутизация позволяет администратору сетевого комплекса задавать те сведения, которые могут сообщаться о закрытых частях сети.

Статический маршрут к сети также достаточен в том случае, если сеть доступна только по одному пути. Такой тип участка сетевого комплекса называется тупиковой сетью.

Эталонная модель *OSI*

При огромном числе компаний, разрабатывающих и производящих оборудование и программы для организации сетей вопрос стандартизации играет важнейшую роль. Возможность работы разнородного оборудования в одной сети обеспечивает стандарт *Ethernet*, разработанный *IEEE*. Использование этого стандарта всеми производителями адаптеров и концентраторов позволяет им выпускать оборудование, способное взаимодействовать с программами и аппаратурой других фирм при работе в одной сети.

Для обеспечения совместимости программ Международная Организация по Стандартизации (*ISO - International Standards Organization*) разработала модель сетевой архитектуры, получившую известность как *OSI*-модель. Модель Взаимодействия Открытых Систем (*Open Systems Interconnect*) описывает структуру сетевых уровней. Не все разработчики программ в точности следуют этой модели, однако она дает основы понимания способов взаимодействия сетевых компонент.

Модель *OSI*

В начале 80-х годов ISO опубликовала модель, предназначенную для оказания помощи разработчикам при объединении различных сетей. Получившая широкую известность модель OSI содержит семь дискретных уровней, каждый из которых обеспечивает выполнение определенной части сетевых функций при обмене данными между компьютерами сети.

Преимущества использования модели *OSI*:

- Деление сложных сетевых операций на легко управляемые уровни.
- Изменения на одном уровне не действуют на другие уровни модели, что позволяет разработчикам приложений специализироваться на определенных и ограниченных задачах.
- Определение стандартных интерфейсов для простой интеграции оборудования различных компаний.

Эталонная модель *OSI* – это описательная схема сети; ее стандарты гарантируют высокую совместимость и способность к взаимодействию различных типов сетевых технологий. Кроме того, она иллюстрирует процесс перемещения информации по сетям. Это концептуальная структура, определяющая сетевые функции, реализуемые на каждом ее уровне. Модель *OSI* описывает, каким образом информация проделывает путь через сетевую среду (например, провода) от одной прикладной программы (например, программы обработки таблиц) к другой прикладной программе, находящейся в другом подключенном к сети компьютере.

Эталонная модель *OSI* делит задачу перемещения информации между компьютерами через сетевую среду на семь менее крупных и, следовательно, более легко разрешимых подзадач.

Каждая из этих семи подзадач выбрана потому, что она относительно автономна и, следовательно, ее легче решить без чрезмерной опоры на внешнюю информацию. Такое разделение на уровни называется *иерархическим представлением*. Каждый уровень соответствует одной из семи подзадач.

Поскольку нижние уровни (с 1 по 3) модели *OSI* управляют физической доставкой сообщений посети, их часто называют *уровнями среды передачи данных* (*media layers*). Верхние уровни (с 4 по 7) модели *OSI* обеспечивают точную доставку данных между компьютерами в сети, поэтому их часто называют *уровнями хост-машины* (*host layers*).

Уровень приложений

Уровень приложений – это самый близкий к пользователю уровень модели *OSI*. Он отличается от других уровней тем, что не предоставляет услуги ни одному другому уровню модели *OSI* и только обслуживает прикладные процессы, находящиеся вне пределов модели *OSI*. Примерами таких прикладных процессов могут служить программы работы с электронными таблицами, текстовые процессоры и т.д.

Уровень приложений идентифицирует и устанавливает доступность предполагаемых партнеров для связи, синхронизирует совместно работающие прикладные программы, а также устанавливает договоренность о процедурах восстановления после ошибок и контроля целостности данных.

Уровень приложений также определяет степень достаточности ресурсов для осуществления предполагаемой связи.

Уровень представлений

Уровень представлений отвечает за то, чтобы информация, посылаемая из уровня приложений одной системы, была читаемой для уровня приложений другой системы. При необходимости уровень представлений преобразовывает форматы данных путем использования общего формата представления информации. Компьютеры настраиваются на их получение; принятые данные преобразуются в формат, пригодный для чтения (например, транслируются из кода *EBCDIC* в код *ASCII*). За счет службы преобразования на уровне представления можно гарантировать, что данные с уровня приложений одной системы попадут на этот же уровень другой системы.

Уровень сеансовый

Сеансовый уровень устанавливает, управляет и завершает сеансы взаимодействия приложений. Сеансы состоят из диалога между двумя или более объектами представления. Сеансовый уровень синхронизирует диалог между объектами уровня представлений и управляет обменом информации между ними. В дополнение к основным функциям сеансовый уровень предоставляет средства для синхронизации участвующих в диалоге сторон, обеспечивает класс услуг и средства формирования отчетов об особых ситуациях, возникающих на сеансовом уровне, а также на уровнях приложений и представлений.

Ниже приведены некоторые протоколы и интерфейсы сеансового уровня:

- *SQL* (*Structured Query Language* – язык структурированных запросов) На

языке *SQL*, разработанном компанией *IBM*, пользователь может в несложной форме определить свои требования к информации, доступ к которой производится на локальных или удаленных системах.

- *RPC* (*Remote Procedure Call* – вызов удаленных процедур) Является простым инструментом переадресации в среде клиент/сервер. Процедуры *RPC* создаются на компьютере клиента и выполняются на сервере.

Уровень транспортный

Транспортный уровень сегментирует и повторно собирает данные в один поток. Если уровень приложений, сеансовый уровень и уровень представлений заняты прикладными вопросами, четыре нижних уровня решают задачу транспортировки данных. Транспортный уровень пытается обеспечить услуги по транспортировке данных, которые изолируют верхние уровни от деталей ее реализации. В частности, заботой транспортного уровня является решение таких вопросов, как выполнение надежной транспортировки данных через многосетевой комплекс. Предоставляя надежные услуги, транспортный уровень обеспечивает механизмы для установки, поддержания и упорядоченного завершения действия виртуальных каналов, обнаружения и устранения неисправностей транспортировки, а также управления информационным потоком (с целью предотвращения переполнения одной системы данными от другой системы).

Уровень сетевой

Сетевой уровень – это комплексный уровень, который обеспечивает соединение и выбор маршрута между двумя конечными системами, которые могут находиться в географически разных сетях.

Уровень канальный

Канальный уровень обеспечивает надежный транзит данных через физический канал. Выполняя эту задачу, канальный уровень решает вопросы физической адресации, топологии сети, дисциплины в канале связи (т.е. каким образом конечная система использует сетевой канал), уведомления об ошибках, упорядоченной доставки кадров, а также вопросы управления потоком данных.

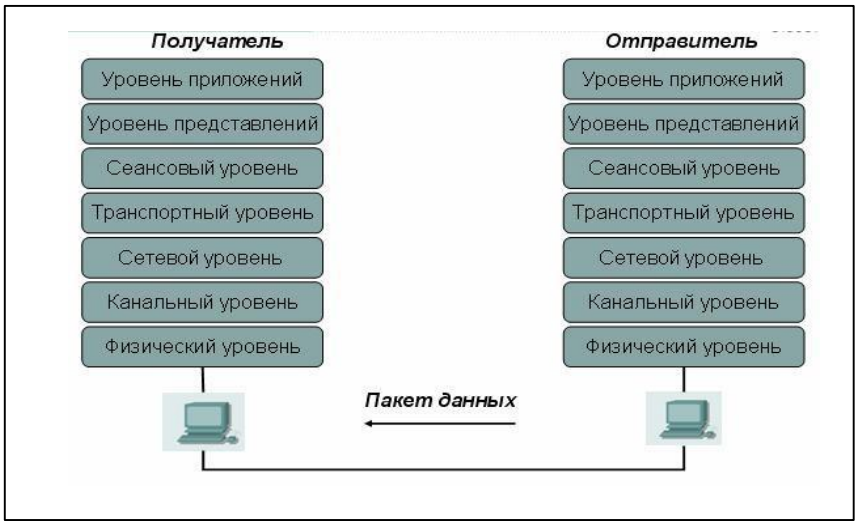
Уровень физический

Физический уровень определяет электротехнические, механические, процедурные и функциональные характеристики активизации, поддержания и деактивизации физического канала между конечными системами. Спецификации физического уровня определяют такие характеристики, как уровни напряжений, временные параметры изменения напряжений, скорости

физической передачи данных, максимальные расстояния передачи информации, физические разъемы, и другие подобные характеристики.

На физическом уровне определен интерфейс между оконечным оборудованием данных (*DTE*) и оконечным оборудованием цепей передачи данных (*DCE*). Если оконечное оборудование цепей передачи данных размещается у провайдера, то оконечным оборудованием данных являются подключенные устройства пользователей. Доступ к устройствам *DTE* наиболее часто осуществляется через модем или модуль обслуживания каналов/данных (*CSU/DSU*).

Инкапсулирование данных

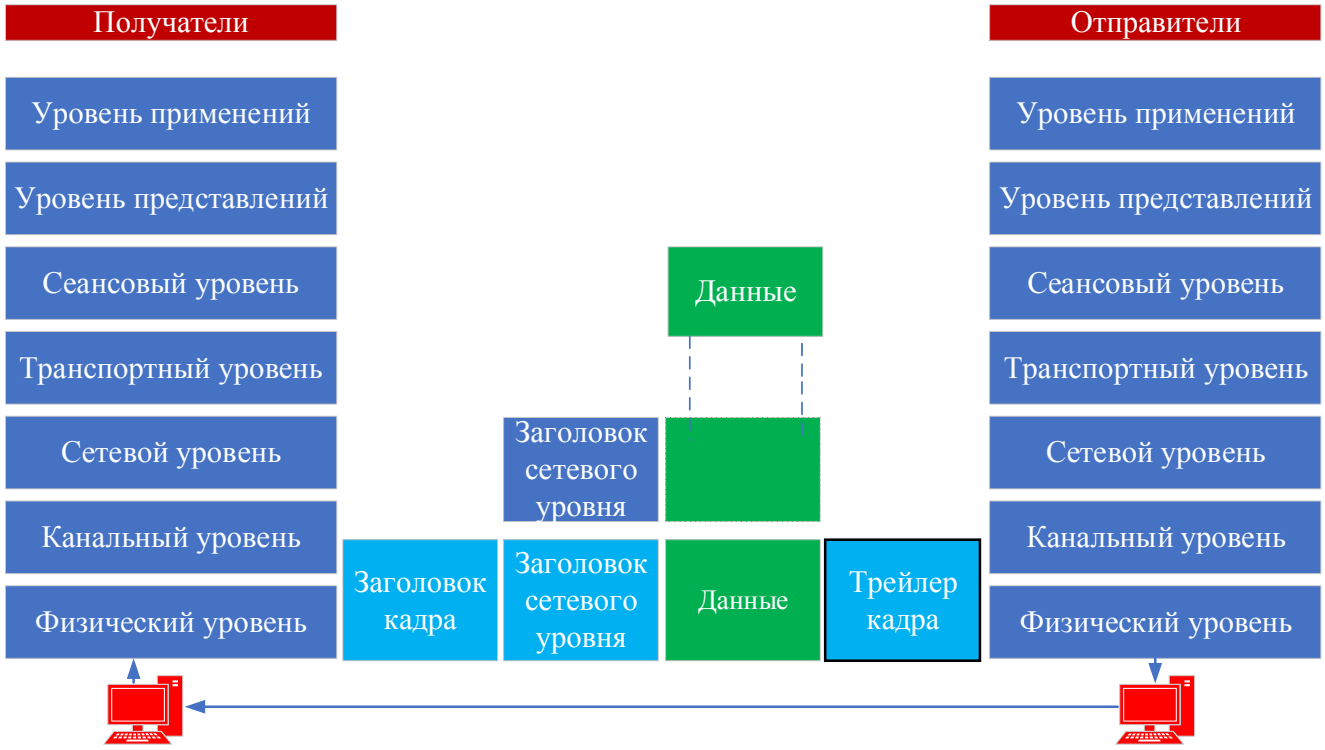


Чтобы понять структуру и принципы функционирования сети, необходимо разобраться как происходит обмен данными в сети от источника к получателю. Информацию, посланную в сеть, называют данными, или пакетами данных. Если один компьютер (источник) хочет послать данные другому компьютеру (получателю), то данные сначала должны быть собраны в пакеты в процессе инкапсуляции; который перед отправкой в сеть погружает их в заголовок конкретного протокола. Этот процесс можно сравнить с подготовкой бандероли к отправке – обернуть содержимое бумагой, вложить в транспортный конверт, указать адрес отправителя и получателя, наклеить марки и бросить в почтовый ящик.

Каждый уровень эталонной модели зависит от услуг нижележащего уровня. Чтобы обеспечить эти услуги, нижний уровень при помощи процесса инкапсуляции помещает *PDU*, полученный от верхнего уровня, в свое поле данных; затем могут добавляться заголовки и трейлеры, необходимые уровню для реализации своей функции. Впоследствии, по мере перемещения данных вниз по уровням модели *OSI*, к ним будут прикрепляться дополнительные заголовки и трейлеры. Например, сетевой уровень обеспечивает поддержку уровня представлений, а уровень представлений передает данные в межсетевую

подсистему.

Задачей сетевого уровня является перемещение данных через сетевой комплекс. Для выполнения этой задачи данные инкапсулируются в заголовок, который содержит информацию, необходимую для выполнения передачи, например, логические адреса отправителя и получателя. Канальный уровень служит для поддержки сетевого уровня и инкапсулирует информацию от сетевого уровня в кадре. Заголовок кадра содержит данные (физические адреса), необходимые канальному уровню для выполнения его функций. Физический уровень служит для поддержки канального уровня. Кадры канального уровня преобразуются в последовательность нулей и единиц для передачи по физическим каналам.



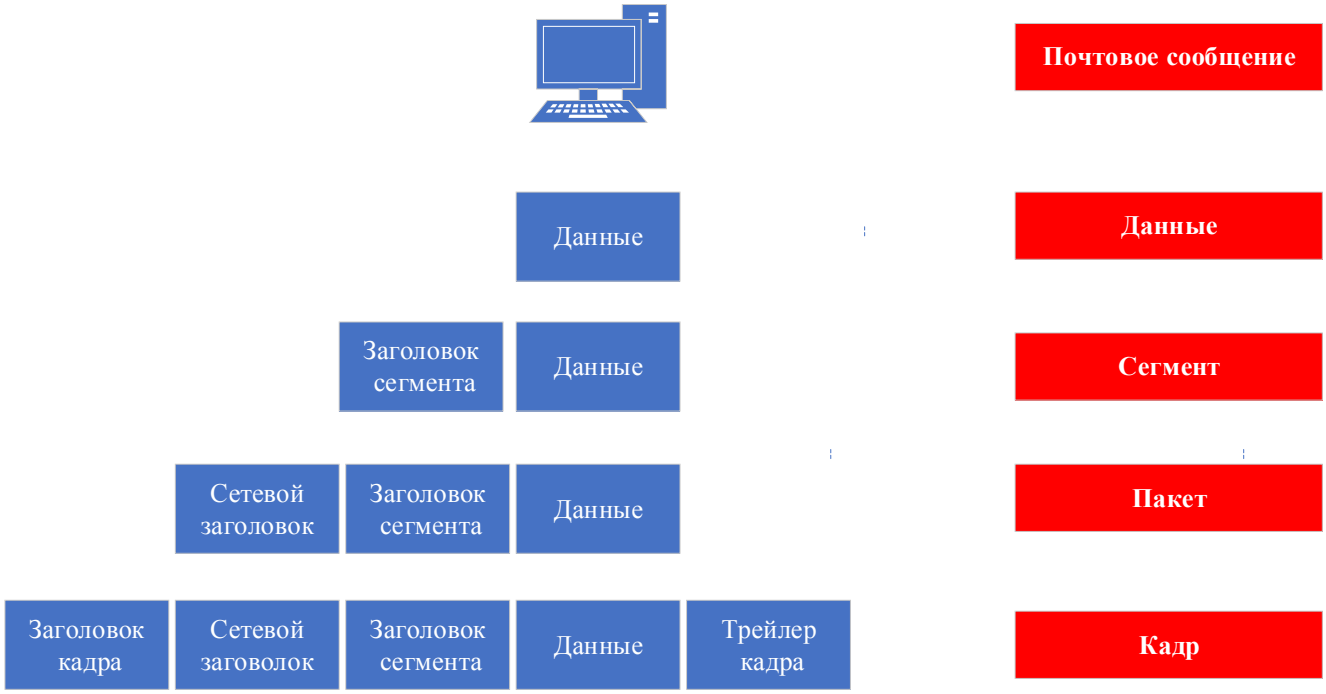
При выполнении сетями услуг пользователям, поток и вид упаковки информации изменяются.

***В инкапсуляции пять этапов преобразования*:**

- Формирование данных. Когда пользователь посылает сообщение электронной почтой, алфавитно-цифровые символы сообщения преобразовываются в данные, которые могут перемещаться в сетевом комплексе. Сетевой уровень оказывает услуги уровню представлений, инкапсулируя **данные** в сетевой заголовок.
- Упаковка данных для сквозной транспортировки. Для передачи через сетевой комплекс данные соответствующим образом упаковываются. Благодаря использованию **сегментов**, транспортная функция гарантирует надежное соединение участвующих в обмене сообщениями хост-машин на обоих концах

почтовой системы.

- Добавление сетевого адреса в заголовок. Данные помещаются в **пакет** или дейтаграмму, которая содержит сетевой заголовок с логическими адресами отправителя и получателя. Эти адреса помогают сетевым устройствам посылать пакеты через сеть по выбранному пути.
- Добавление локального адреса в канальный заголовок. Каждое сетевое устройство должно поместить пакеты в **кадр.** Кадры позволяют взаимодействовать с ближайшим непосредственно подключенным сетевым устройством в канале. Каждое устройство, находящееся на пути движения данных по сети, требует формирования кадров для соединения со следующим устройством.
- Преобразование в последовательность битов для передачи. Для передачи по физическим каналам (обычно по проводам) кадр должен быть преобразован в **последовательность единиц и нулей.**



Сетевые стандарты *Ethernet* и *IEEE* 802.3

Ethernet был разработан в 1970 году и является на сегодняшний день наиболее популярным стандартом.

Ethernet стал основой для спецификации IEEE 802.3, которая была выпущена в 1980. На сегодняшний день *Ethernet* и *IEEE* 802.3 являются наиболее распространенными стандартами локальных вычислительных сетей. Технология *Ethernet* дает возможность устройствам коллективно пользоваться одними и теми же ресурсами, т.е. все устройства могут пользоваться одной средой доставки. Средой доставки называется метод передачи и приема данных. Например, электронные данные могут передаваться по медному кабелю, по тонкому или

толстому коаксиальному кабелю, по беспроводным линиям связи и т. д.

Стандарты *Ethernet* и *IEEE* 802.3 определяют локальные сети с шинной топологией, работающие в монополосном режиме со скоростью передачи 10 Мбит/с. Такие ЛВС называют 10*Base*.

Технология Ethernet:

- 10*Base*2 – тонкий *Ethernet*; допускает протяженность сетевых сегментов на коаксиальном кабеле до 185 метров.
- 10*BaseS* – толстый *Ethernet*; допускает протяженность сетевых сегментов на коаксиальном кабеле до 500 метров.
- 10*BaseT* – использует для передачи кадров недорогой кабель на основе витой пары.

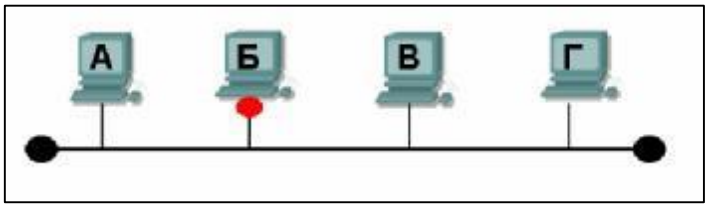
Стандарты 10*BaseS* и 10*Base*2 обеспечивают доступ к нескольким станциям в одном сегменте ЛВС. Станции подключаются к сегменту с помощью кабеля, который одним концом соединяется с интерфейсом блока подключения (*attachment unit interface, AUI*) на станции, а другим – с трансивером, подключаемым к коаксиальному кабелю *Ethernet*.

Поскольку стандарт 10*BaseT* обеспечивает доступ только для одной станции, то в локальных сетях на базе 10*BaseT* станции почти всегда подключаются к концентратору или сетевому коммутатору. При подобной конфигурации принято считать, что концентратор или сетевой коммутатор относится к тому же сегменту, что и подключенные к нему станции

Принцип работы сеть *Ethernet 802.3*

В сети *Ethernet* данные, посылаемые одним узлом, проходят через весь сегмент. По мере движения данные принимаются и анализируются каждым узлом. Когда сигнал достигает конца сегмента, он поглощается специальным оконечным элементом. Это необходимо для того, чтобы предотвратить движение сигнала в обратном направлении. В каждый отдельный момент времени в локальной сети возможна только одна передача. Например, в сети с линейной шинной топологией пакет данных передается от станции Б к станции Г. Этот пакет принимается всеми станциями.

Станция Г распознает свой адрес и обрабатывает кадр. Станции А и В не распознают свои *MAC*-адреса и игнорируют кадр.



Множественный доступ с контролем несущей и обнаружением конфликтов

Сегодня термин стандартный *Ethernet* чаще всего применяется для описания всех ЛВС, использующих технологию *Ethernet* (технологию коллективного использования среды передачи данных), которая в общем случае удовлетворяет

требованиям спецификаций *Ethernet,* включая спецификации стандарта *IEEE* 802.3. Чтобы использовать принцип коллективной работы со средой передачи данных, в *Ethernet* применяется протокол множественного доступа с контролем несущей и обнаружением конфликтов (*carrier sense multiple access/collision detection, CSMA/CD*), Использование протокола *CSMA/CD* позволяет устройствам договариваться о правах на передачу. *CSMA/CD* является методом доступа, который позволяет только одной станции осуществлять передачу в среде коллективного использования. Задачей стандарта *Ethernet* является обеспечение качественного сервиса доставки данных. Не все устройства могут осуществлять передачу на равных правах в течение всего времени, поскольку это может привести к возникновению конфликтов. Однако стандартные сети Ethernet, использующие протокол *CSMA/CD,* учитывают все запросы на передачу и определяют, какие устройства могут передавать в данный момент и в какой последовательности смогут осуществлять передачу все остальные устройства, чтобы все они получали адекватное обслуживание.

Перед отправкой данных узел "прослушивает" сеть, чтобы определить, можно ли осуществлять передачу, или сеть сейчас занята. Если в данный момент сеть никем не используется, узел осуществляет передачу. Если сеть занята, узел переходит в режим ожидания. Возникновение конфликтов возможно в том случае, если два узла, "прослушивая" сеть, обнаруживают, что она свободна, и одновременно начинают передачу. В этом случае возникает конфликт, данные повреждаются и узлам необходимо повторно передать данные позже. Алгоритмы задержки определяют, когда конфликтующие узлы могут осуществлять повторную передачу. В соответствии с требованиями *CSMA/CD,* каждый узел, начав передачу, продолжает "прослушивать" сеть на предмет обнаружения конфликтов, узнавая таким образом о необходимости повторной передачи.

Метод *CSMA/CD* работает следующим образом: если узел хочет осуществить передачу, он проверяет сеть на предмет того, не передает ли в данный момент другое устройство. Если сеть свободна, узел начинает процесс передачи. Пока идет передача, узел контролирует сеть, удостоверяясь, что в этот же момент времени не передает никакая другая станция. Два узла могут начать передачу почти одновременно, если обнаружат, что сеть свободна. В этом случае возникает конфликт. Когда передающий узел узнает о конфликте, он передает сигнал "Наличие конфликта". После этого все передающие узлы прекращают отправку кадров на выбираемый случайным образом отрезок времени, называемый временем задержки повторной передачи. По истечении этого периода осуществляется повторная передача. Если последующие попытки также заканчиваются неудачно, узел повторяет их до 16 раз, после чего отказывается от передачи. Время задержки для каждого узла разное. Если различие в длительности этих периодов задержки достаточно велико, то повторную передачу узлы начнут уже не одновременно. С каждым последующим конфликтом время задержки удваивается, вплоть до десятой попытки, тем самым уменьшая вероятность возникновения конфликта при повторной передаче.

IP-адресация

Каждый компьютер, независимо от того, подключен он к сети или нет, имеет уникальный физический адрес. Не существует двух одинаковых физических адресов. Физический адрес (или *MAC*-адрес) зашит на плате сетевого адаптера. Каждая плата сетевого адаптера, который работает на канальном уровне эталонной модели *OSI,* имеет свой уникальный *MAC*-адрес. В сети, когда одно устройство хочет переслать данные другому устройству, оно может установить канал связи с этим другим устройством, воспользовавшись его *MAC*-адресом. Отправляемые источником данные содержат *MAC*- адрес пункта назначения. По мере продвижения пакета в среде передачи данных сетевые адаптеры каждого из устройств в сети сравнивают *MAC*-адрес пункта назначения, имеющийся в пакете данных, со своим собственным физическим адресом. Если адреса не совпадают, сетевой адаптер игнорирует этот пакет, и данные продолжают движение к следующему устройству. Если же адреса совпадают, то сетевой адаптер делает копию пакета данных и размешает ее на канальном уровне компьютера. После этого исходный пакет данных продолжает движение по сети, и каждый следующий сетевой адаптер проводит аналогичную процедуру сравнения.

Перед тем как отправить пакет данных ближайшему устройству в сети, передающее устройство должно знать *MAC*-адрес назначения. Поэтому механизм определения местоположение компьютеров в сети является важным компонентом любой сетевой системы. В зависимости от используемой группы протоколов применяются различные схемы адресации.

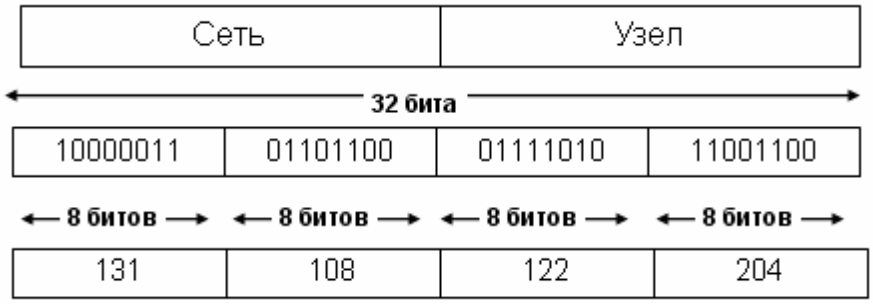
В сетях используются две схемы адресации:

- *MAC*- адресация.
- *IP*-адресация.

Как следует из названия, *IP*-адресация базируется на протоколе *IP* (*Internet Protocol*). Каждая ЛВС должна иметь свой уникальный *IP*-адрес, который является определяющим элементом для осуществления межсетевого взаимодействия в глобальных сетях. В *IP*- сетях конечная станция связывается с сервером или другой конечной станцией. Каждый узел имеет *IP*-адрес, который представляет собой уникальный 32-битовый логический адрес. *IP*-адресация существует на уровне 3 (сетевом) эталонной модели *OSI*. В отличие от *MAC*-адреса, которые обычно существуют в плоском адресном пространстве, *IP*-адреса имеют иерархическую структуру.

Достаточно трудно запомнить число, состоящее из 8 цифр, не говоря уже о числах из 32 цифр, которые используются в *IP*-адресах. Поэтому для обозначения 32-битовых чисел в *IP*-адресах используются десятичные числа. Это называется представлением в десятичной форме с разделением точками.

IP адрес записывается в виде четырех десятичных номеров, разделенных точками, и каждая из четырех частей называется октетом.



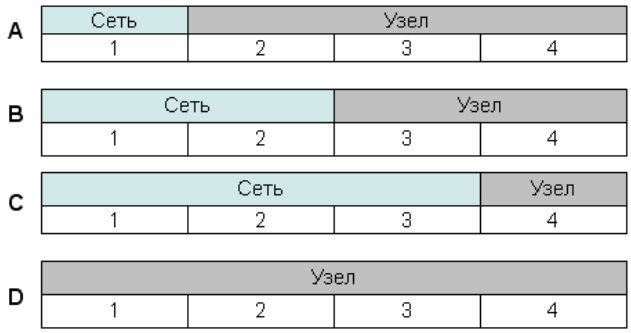
Классы *IP*-адресов

Для того чтобы каждый сетевой адрес был уникальным и отличался от любого другого номера, организация *American Registry for Internet Numbers* (Американский реестр *Internet*- номеров, *ARIN*) выделяет компаниям блоки *IP*-адресов в зависимости от размера их сетей. Каждый *IP*-адрес состоит из двух частей: номера сети и номера хоста. Сетевой номер идентифицирует сеть, к которой подключено устройство. Номер хоста идентифицирует устройство в этой сети.

ARIN определяет три класса *IP*-адресов. Класс *A* составляют *IP*-адреса, зарезервированные для правительственных учреждений, класс *B* – *IP*-адреса для компаний среднего уровня и класс *C* –- для всех остальных организаций.

Зарезервированные классы сетей

На самом деле существует пять классов сетевых адресов. Но только три из них - классы *A, B* и *C* – используются коммерчески. Два других класса сетевых адресов зарезервированы.



Максимально возможное значение каждого октета *IP*-адреса равно 255. Следовательно, это десятичное число могло бы быть присвоено первому октету сети любого класса. На практике применяются только числа до 223. Возникает вопрос: почему при максимально допустимом значении 255 для каждого октета используются только числа до 223?

Недостающие два адреса резервируется для экспериментальных целей и потребностей групповой адресации. Эти номера не могут быть присвоены сетям. Поэтому в первом октете *IP*-адресов значения с 224 по 255 для решения сетевых задач не используются.

Кроме этих зарезервированных адресов резервируются также все *IP*-адреса, у которых в той части адреса, которая обозначает адрес хост-машины, содержатся только нули или единицы.

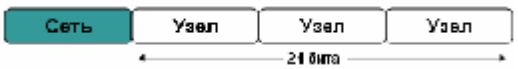
В приведенных ранее примерах *IP*-адреса использовались только по отношению к устройствам, подключенным к сети. Иногда необходимо обратиться ко всем устройствам в сети, или, другими словами, к самой сети. Однако довольно сложно выписать адреса всех устройств в сети. Можно было бы использовать только два адреса с дефисом между ними, для того чтобы показать, что обращение осуществляется ко всем устройствам в заданном диапазоне чисел, но и это достаточно сложно. Вместо этого придуман более простой метод обращения ко всей сети. В соответствии с соглашением, в схемах *IP*- адресации любой *IP*-адрес, который заканчивается всеми двоичными нулями, резервируется для адреса этой сети. Примером адреса сети класса *A* может быть *IP*-адрес 113.1.1.1.

Это объясняется тем, что оба октета назначаются *ARIN* и обозначают номер сети. Только два последних октета содержат нули. Это связано с тем, что числа в этих октетах обозначают номера хостов, зарезервированные для устройств, подключаемых к сети. Следовательно, для того, чтобы обратиться ко всем устройствам в этой сети, т.е. к самой сети, сетевой адрес должен иметь нули в двух последних октетах.

Процесс, в ходе которого источник отправляет данные всем устройствам в сети, называется широковещанием. Для того чтобы все устройства в сети обратили внимание на широковещание, должен использоваться такой *IP*-адрес, который смогли бы распознать и признать своим все устройства в сети. Например, для сети 176.10.0.0, адресом широковещания может быть адрес 176.10.255.255.

Адреса класса А

В *IP*-адресе сетей класса *A* первый байт занимает адрес сети, а в трех последующих байтах размещается адрес узла. Формат *IP*-адреса сети класса *A*:



Например, в *IP*-адресе 49.22.102.70 адрес сети равен 49, а адрес узла - 22.102.70. Каждая машина этой сети должна иметь адрес сети, равный 49.

Адрес сети класса *A* имеет длину 1 байт, причем его первый бит зарезервирован, но доступны оставшиеся семь разрядов. Это означает, что можно создать не более 128 сетей класса *A*. Почему? Потому что каждый из семи оставшихся битов может принимать значение 0 или 1, т.е. существует 27 или 128 различных комбинаций. Было решено, что нулевой адрес сети (0000 0000) резервируется для обозначения маршрута, выбранного по умолчанию. Однако из-за того, что нулевой адрес зарезервирован, диапазон становится уже: от 1 до 127. В результате реальное число сетей класса *A* равно 128-2, т.е. 126.

Под адрес узла в IP-адресе сетей класса А отведено 3 байта (24 разряда). В них можно разместить 224 или 16 777 216 различных двоичных комбинаций

или адресов узлов. Поскольку адреса, состоящие только из нулей и только из единиц, зарезервированы, точное число узлов в сети класса А составляет 224 - 2=16 777 214.

Адреса класса *B*

В *IP*-адресе сетей класса *B* первые два байта занимает адрес сети, а в двух последующих байтах размещается адрес узла. Формат *IP*-адреса сети класса В:

Например, в IP-адресе 172.16.30.56 адрес сети равен 172.16, а адрес узла – 30.56. Для адреса сети, состоящего из 16 разрядов, имеется 216 возможных комбинаций. Адрес сети класса *B* начинается с комбинации 10, поэтому свободными для формирования адреса остаются лишь 14 бит; это означает, что может существовать 214 или 16 384 сетей класса В. Под адрес узла в *IP*-адресе сетей класса В отведено 2 байта. Поскольку адреса, состоящие только из нулей и только из единиц, зарезервированы, точное число узлов в сети класса В равно 216 - 2=65 534.

Адреса класса *C*

Первые три байта, в *IP*-адресе сетей класса *C* занимает адрес сети, и всего один байт остается для адреса узла.

Формат *IP*-адреса сети класса *C*:

Например, в *IP*-адресе 192.168.100.102 адрес сети равен 192.168.100, а адрес узла –102.

Первые три разряда адреса сети класса С занимает комбинация 110. Поэтому для формирования адреса остается лишь 24 - 3= 21 разряд. Таким образом, может существовать 221 или 2 097 152 сетей класса С. Под адрес узла в *IP*-адресе сетей класса *C* отведен 1 байт. Следовательно, в каждой сети класса *C* может быть 28 - 2=254 узла.

Выделение подсетей

Выше мы обсудили присвоение адресов и определение количества хостов в сетях класса *A, B* и *C*. Однако таким способом можно указать только одну сеть. Что делать, если получен один сетевой адрес, но нужно создать на его основе несколько сетей? Следует использовать выделение подсетей (*subnetting*), т.е. деление одной большой сети на несколько маленьких.

4. Сетевое администрирование

Управление компьютерной сетью

В начале 1980-х годов персональные компьютеры стали объединять в сети для обмена данными и совместного использования файлов и ресурсов. К середине 1980-х годов эти сети становятся крупными и сложными. Для управления ими создаются отделы информационного обеспечения.

Управление сетью (*Network management*) – целенаправленное воздействие на сеть, осуществляемое для организации её функционирования по заданной программе. Оно включает следующие процедуры:

- включение и отключение системы, каналов передачи данных, терминалов;
- диагностика неисправностей;
- сбор статистики;
- подготовка отчётов и т.п.

С точки зрения модели *OSI* управление сетью подразделяется на управление:

- конфигурацией;
- отказами
- безопасностью;
- трафиком;
- учётом.

Традиционные методы управления основаны на использовании правил. Они предписывают системе управления в компьютерной сети предпринимать определённые действия (например, выдать предупреждающее сообщение на управляющую консоль) при наступлении определённых событий (превышение интенсивностью трафика заранее определённого порогового значения и др.).

Приемлемая в небольших сетях, методология управления на основе правил сталкивается с множеством препятствий в крупных сетях: сетях вычислительных центров и корпоративных информационных сетях (ИС). Основная трудность обусловлена тем, что функционирование мощной вычислительной среды может описываться многими тысячами параметров.

Корпоративная сеть (сеть масштаба предприятия, *Enterprise network*) – сеть смешанной топологии, в которую входят несколько локальных вычислительных сетей. Корпоративная сеть объединяет филиалы корпорации и является собственностью предприятия.

Сеть вычислительных центров – совокупность взаимодействующих вычислительных центров (узлов), объединённых каналами связи для наиболее полного обеспечения потребности пользователей (абонентов) в выполнении информационно-вычислительных работ.

Пользователь/Посетитель *(User; Visitor):*

1) абонент (клиент) сетевого ресурса;

2) посетитель сервера (сайта, портала) и пользователь доступного ему информационного ресурса в сети.

Сервер *(Server)* – компьютер, подключенный к сети, или выполняющаяся на нём программа, предоставляющие клиентам доступ к общим ресурсам и управляющие этими ресурсами.

Система управления сетью *(Network management system)* – аппаратные и (или) программные средства, применяемые для мониторинга и управления узлами сети. Программное обеспечение системы управления сетью состоит из агентов, локализующихся на сетевых устройствах и передающих информацию сетевой управляющей платформе.

Платформа управления сетью (*Network management platform****)*** – комплекс программ, предназначенных для управления сетью и входящими в неё системами. Для работы с платформой администратору предоставляется одна или несколько абонентских систем (консолей). Обычно платформа создаётся на базе протокола *SNMP*. Платформа обеспечивает:

- контроль работы устройств и состояния кабелей;
- контроль деловых процедур;
- контроль других аспектов функционирования сети.

Чтобы компьютерная сеть могла эффективно выполнять свои функции, необходимо централизованно контролировать состояние основных её элементов, выявлять и разрешать возникающие проблемы, выполнять анализ производительности и планировать развитие сети и др. Эти виды работ являются основными задачами администрирования сетей.

Системное и сетевое администрирование

Администрирование – процедуры управления, регламентирующие некоторые процессы или их часть. Как правило, оно фиксирует и руководит процессами и ситуациями, нуждающимися в ограничениях или целевом управлении. Построение компьютерных сетей вызвало необходимость управления (администрирования) ими и созданными на их основе компьютерными вычислительными и информационными системами. В результате появилось системное администрирование.

Системное администрирование

Основной целью системного администрирования является приведение сети в соответствие с целями и задачами, для которых она предназначена. Достигается эта цель путём управления сетью, позволяющего минимизировать затраты времени и ресурсов, направляемых на управление системой, и в тоже время максимизировать доступность, производительность и продуктивность системы.

История системного администрирования насчитывает несколько десятилетий. Задачи управления вычислительными комплексами (системами) возникли сразу после появления самих этих комплексов. Доминировавшая до конца 1980-х годов централизованная вычислительная модель типа “мэйнфрейм-терминалы” непосредственно проецировалась на архитектуру средств администрирования, которую относят к категории системного. Такое решение означало существование единого образа вычислительной среды. В подобных средствах администрирования задачи управления сводились к контролю за функционированием отдельных компонентов, причём во многих случаях он заключался просто в сборе данных о ресурсах вместо действительного управления их работой. Такой тип управления нельзя отнести к сетевому администрированию в строгом смысле этого слова.

В начале 1990-х годов широкое распространение распределённых архитектур “клиент-сервер” вызвало перемены в управлении информационными системами, сменившими безраздельное господство хост-компьютеров. Вместо однородной среды администраторам пришлось иметь дело с многообразием ресурсов, включая компьютерные и программные платформы, а также сетевое оборудование. Такое положение потребовало решения новых административных задач: учёта распределённых ресурсов, электронного распространения ПО и контроля лицензий, анализа трафика и управления пропускной способностью сети, перераспределения серверной нагрузки, отслеживания состояния отдельных настольных систем и другое, отсутствовавших в классической централизованной модели. В эту среду не переносились приложения администрирования, функционировавшие на мэйнфреймах, и производителям пришлось создавать новое управляющее ПО. Всё это способствовало появлению сетевого администрирования.

Сетевое администрирование

Сетевое администрирование (*Network Management*) возникает, когда у администратора сети появляется потребность и возможность оперировать единым представлением сети, как правило, это относится к сетям со сложной архитектурой. При этом осуществляется переход от управления функционированием отдельных устройств к анализу трафика в отдельных участках сети, управлению её логической конфигурацией и конкретными рабочими параметрами, причём все эти операции целесообразно выполнять с одной управляющей консоли. Задачи, решаемые в данной области, разбиваются на две группы:

1. Контроль за работой сетевого оборудования,
2. Управление функционированием сети в целом.

Конечной целью управления сетью является достижение параметров функционирования ИС, соответствующих потребностям пользователей. Пользователи оценивают работу ИС не по характеристикам сетевого трафика, применяемым протоколам, времени отклика серверов на запросы определённого типа и особенностям выполняемых сценариев управления, а по поведению приложений, ежедневно запускаемых на их настольных компьютерах.

Общая тенденция в мире сетевого и системного администрирования – перенос акцентов с контроля за отдельными ресурсами или их группами, с управления рабочими характеристиками ИС на максимальное удовлетворение запросов конечных потребителей информационных технологий способствовала появлению концепции динамического администрирования.

Такой подход предполагает, прежде всего, наличие средств анализа поведения пользователей, в ходе которого выявляют их предпочтения и проблемы, возникающие в повседневной работе.

Результаты, полученные на этом этапе, должны послужить отправной точкой для активного управления взаимодействием между основными объектами администрирования – пользователями, приложениями и сетью. Эти факторы дают основание полагать, что на смену сетевому и системному администрированию придёт управление приложениями и качеством сервиса, независящее от используемых вычислительных платформ или сетей.

Эволюция концепций администрирования коснулась не только архитектуры систем. Новые проблемы, возникшие в распределённых средах, привели к тому, что на некоторое время сетевое управление стали рассматривать как главную заботу администраторов ИС. Ситуация изменилась, когда число распределённых приложений и баз данных, функционирующих в сети, превысило пороговое значение. При этом возросла роль системного администрирования, и неизбежным оказался процесс интеграции системного и сетевого администрирования.

Интегрированная система управления сетью *(Integrated network management system, INMS)* – система управления, обеспечивающая объединение функций, связанных с анализом, диагностикой и управлением сетью.

Таким образом, эволюция средств и систем администрирования непосредственно связана с развитием основных информационных технологий.

Проекты развития административных механизмов обычно включают в себя задачи постановки стратегического управления, разработки политики информационного обеспечения и доступа к информационным ресурсам, а также программно-аппаратным устройствам, системам и комплексам, постановки и развития системы, совершенствование непрерывного управления.

Трудно говорить о том, по какому пути – интеграционному или дезинтеграционному – пойдёт развитие сетей. Ряд экспертов предполагает, что на смену сетевому и системному администрированию придёт управление приложениями и качеством сервиса, безотносительно к используемым вычислительным платформам или сетям. В любом случае управление сетями осуществляют сетевые администраторы (администраторы сетей).

Цели и задачи администратора сети

Администратор сети – специалист, отвечающий за нормальное функционирование и использование ресурсов автоматизированной системы и (или) вычислительной сети.

Администрирование информационных систем включает следующие цели:

1. Установка и настройка сети.
2. Поддержка её дальнейшей работоспособности.
3. Установка базового программного обеспечения.
4. Мониторинг сети.

В связи с этим администратор сети должен выполнять следующие задачи:

- Планирование системы.
- Установка и конфигурация аппаратных устройств.
- Установка программного обеспечения.
- Установка сети.
- Архивирование (резервное копирование) информации.
- Создание и управление пользователями.
- Установка и контроль защиты.
- Мониторинг производительности.

Обеспечение работоспособности системы требует и осуществления профилактических мероприятий. Администратор должен обеспечивать удовлетворение санкционированных запросов пользователей.

Очевидно, что эффективно выполнять все эти функции и задачи, особенно в сложных крупных компьютерных сетях, человеку весьма затруднительно, а порой и невозможно. Успешное администрирование, особенно сложными компьютерными сетями, реализуется путём применения новейших средств и систем автоматизации этих процессов.

Автоматизация управления сетью

Автоматизированная информационная система *(Automated information system, AIS*) – совокупность программных и аппаратных средств, предназначенных для хранения и (или) управления данными и информацией и производства вычислений. Следовательно, автоматизированная информационная система (АИС) является частью любого административного механизма – платформой управления и сетевой службой.

Платформа управления сетью *(Network management platform)* – комплекс программ, предназначенных для управления сетью и входящими в неё системами.

Сетевая служба использует сервис, предоставляемый областью взаимодействия, и обеспечивает связь прикладных процессов, расположенных в различных абонентских системах сети.

Система административных регламентов и информационная система являются затратной частью системы управления, но их отсутствие не гарантирует качество, оперативность и эффективность управления. При этом технологии и инструменты являются более стабильной компонентой, чем системы управления. Изменения стратегии, политики, методики, исполнителей обычно приводят к изменению системы управления и информационной системы. При этом инструмент, например, существующая АИС, может обеспечить решение новых задач, порой с минимальной дополнительной её настройкой. Смена инструмента

обычно приводит к изменению работы системы управления. Поэтому при целостном, целенаправленном формировании и функционировании системы управления, следует осуществлять одновременное развитие АИС и административных механизмов управления ей.

Очевидно, что управление сетью, как правило, целесообразно осуществлять с одного рабочего места. Потребность в контроле за сетью с одной управляющей станции способствовала появлению различных архитектур платформ и приложений администрирования. Наибольшее распространение среди них получила двухуровневая распределённая архитектура “менеджер–агенты”. Программа-менеджер функционирует на управляющей консоли, постоянно взаимодействуя с модулями-агентами, запускаемыми в отдельных устройствах сети. На агентов в такой схеме возлагаются функции сбора локальных данных о параметрах работы контролируемого ресурса, внесение изменений в его конфигурацию по запросу от менеджера, предоставление последнему административной информации. Однако её применение в реальной сетевой среде приводит к возрастанию объёмов служебного трафика и, как следствие, снижению эффективной пропускной способности, доступной приложениям.

В качестве частичного решения проблемы исчерпания пропускной способности предлагается трёхуровневая схема, в которой часть управляющих функций делегируется важнейшим сетевым узлам. Инсталлированные в этих узлах программы-менеджеры через собственную сеть агентов управляют работой “подотчётных” им устройств и одновременно сами выступают в роли агентов по отношению к основной программе-менеджеру (менеджеру менеджеров), запущенной на управляющей станции. В результате основная часть служебного трафика оказывается локализованной в отдельных сетевых сегментах, поскольку “общение” локальных менеджеров с административной консолью осуществляется только тогда, когда в этом действительно возникает необходимость.

Одна из современных идей совершенствования технологий администрирования сетью заключается в сведении к минимуму роли человека в процессе администрирования ИС. Она подразумевает создание ПО, необходимого для администрирования ИС, например, совмещение контроля защиты, управления пользователями, маршрутизации, резервного копирования информации в случае сбоев и т.д. Администрирование сети в этом случае осуществляет программа, настраиваемая администратором сети. Такое решение значительно облегчает процесс администрирования, поскольку настройка одной программы намного легче, чем настройка всей сети и всех приложений, связанных с работой в сети.

Другое предложение базируется на использовании беспроводных сетей с высокой скоростью передачи информации, например, на основе информации, передаваемой светом, что позволяет избежать проблем, связанных с самим физическим строением сети и значительно увеличить скорость передачи информации, а также избежать ряда проблем, имеющихся у проводных сетей.

Еще одна идея заключается в создании для администрирования информационных систем интеллектуального компьютера – нейрокомпьютера.

Такое решение позволяет свести на нет роль человека в администрировании информационных систем, в то же время добиться максимального быстродействия сетей, и полного их соответствия заданным целям.

Многопользовательские информационные системы

Увеличение размеров сети приводит как к увеличению количества устройств, обеспечивающих её функционирование, так и к увеличению их сложности и числа пользователей сети. В различных сетях эта проблема может осложняться гетерогенностью программного и аппаратного обеспечения. При этом такие информационные системы и среды являются многопользовательскими.

Многопользовательская среда – компьютерная сетевая программа, создающая среду для реализации различных типов поведения различных пользователей.

Сложность организации гетерогенной компьютерной сети вынуждает администратора определять и использовать устройства для соединения двух сетей. Всё шире для этих целей используют специальные устройства: повторители, мосты маршрутизаторы.

Повторители – самые простые устройства для соединения сетей (в т.ч. ЛВС), предназначенные для увеличения длины сегмента сети (сетевого кабеля). Так как повторители ретранслируют в другую сеть все принимаемые пакеты или кадры, то загрузка общей сети аддитивно возрастает.

Мосты – достаточно эффективное средство объединения разных сетей в простой сетевой структуре. Они позволяют локализовать нагрузку в отдельных сегментах сети, не пропуская пакеты или кадры, адресованные станциям данного сегмента, в другие сегменты, что приводит к значительному повышению эффективности использования всей сети за счёт снижения общего трафика. В сетях с относительно простой топологией достаточно легко гарантировать существование одного пути между любыми двумя устройствами. Когда количество соединений велико или сетевые связи становятся более сложными, увеличивается вероятность возникновения дублирующих маршрутов (маршрутизация) между устройствами, которые могут приводить к возникновению паразитных циклов (*active loops*). Хотя администраторам с целью создания избыточных связей может потребоваться формировать и дублирующие маршруты.

Маршрутизатор – специальный компьютер, распознающий различные протоколы и способный правильно направлять пакеты информации из одной сети в другую. Маршрутизаторы более гибкие устройства, чем мосты. Они могут различать пути в зависимости от цены, скорости и задержки в сети; использовать все активные пути, имеющиеся в сети, а также обеспечивать разграничение потоков данных между разными подсетями. Маршрутизаторы облегчают управление большими сетями; поддерживают любую топологию и обеспечивают простой процесс настройки при увеличении размеров и сложности сети.

В отличие от мостов маршрутизаторы работают с логическими идентификаторами каждого сегмента сети. В связи с этим межсетевое

взаимодействие, основанное на маршрутизаторах, позволяет объединить множество логически различных подсетей, в принципе являющихся независимыми административными доменами. Маршрутизаторы являются многопротокольными. В отличие от мостов, они имеют ПО, позволяющее реализацию соответствующего протокола и работающее с более полной информацией, сохраняемой в БД маршрутизатора. БД маршрутизатора называется таблицей маршрутизации и отличается от БД моста. Принципиальное различие состоит в том, что таблица маршрутизации включает информацию о путях (маршрутах), пройденных каждым пакетом по сети от отправителя до получателя.

Существуют устройства, совмещающие функции мостов и маршрутизаторов. ***Мосты-маршрутизаторы (****bridge/router)* – устройства, позволяющие совместить преимущества мостов и маршрутизаторов. Как правило, мост-маршрутизатор реализует полные функции маршрутизации согласно одному или нескольким протоколам и действует как мост для всех других протоколов. Для повышения скорости передачи информации по межсетевому соединению многие мосты и маршрутизаторы используют алгоритмы сжатия данных, дающие выигрыш в случае использования низкоскоростных или сильно загруженных линий связи. При этом скорость передачи данных в межсетевом соединении более 64 Кбит/с не приносит выгод от сжатия данных, так как в этом случае может потребовать больше времени, чем простая передача данных без сжатия.

Многопользовательские объектно-ориентированные среды

Многопользовательские объектно-ориентированные среды (MOOs) – это основанные на тексте среды, установленные на персональных компьютерах для осуществления коммуникации в реальном режиме времени между двумя и более удалёнными друг от друга участниками. Пользователи общаются между собой через Интернет, применяя доступное в сети специальное программно-инструментальное обеспечение, включая осуществляемую в реальном режиме времени аудиосвязь или текстовый чат. Изначально MOOs использовались для неформального общения и профессионального обучения на расстоянии. Работа в многопользовательской среде в первую очередь связана с присутствием в ней большого количества различных пользователей.

Пользователь информационной системы (*Information system user)* – лицо, группа лиц или организация, пользующиеся услугами информационной системы для получения информации или решения других задач.

Пользователей можно разделить на две категории:

1. Администраторы – пользователи, совершающие программные настройки и установки сети и устройств в ней, просматривающие все сообщения системы, изменяющие её свойства и др.

2. Обычные пользователи – пользователи, которые имеют доступ к сетевым ресурсам и устройствам, определяемым администраторами.

Особенности работы в многопользовательских средах

Организацию распределённой структуры в многопользовательской сетевой среде Интернета можно выполнить, используя собственные возможности организации (аренда физической сети передачи данных), или применяя существующие сети российских Интернет-провайдеров. При принятии решения, как правило, учитывается множество факторов, главные из которых могут быть не связаны с техническими решениями. Интернет сыграл решительную роль в развитии распределённых вычислений. Он существенно расширил возможности применения распределённых приложений, позволяя подключать удалённых пользователей и делая функции приложения доступными всем и везде.

Сети с выделенным сервером и сети типа "клиент/сервер"

Сети с выделенным сервером опираются на специализированные компьютеры, называемые серверами, представляющие централизованные хранилища сетевых ресурсов и объединяющие централизованное обеспечение безопасности и управление доступом. Сети с выделенным сервером – это скорее обозначение одноранговой сети с ярко выраженным администрированием. Сетевые протоколы задают однаранговую сеть, а прикладное программное обеспечение превращает её в сеть с выделенным сервером.

Одноранговая сеть – информационная сеть, в которой все рабочие станции могут выступать по отношению к другим рабочим станциям сети как серверы.

Она использует технологии распределённых коммуникаций, при этом все узлы работают под управлением одной операционной системы. В отличие от сетей с выделенным сервером, одноранговые сети не имеют централизованного обеспечения безопасности и управления. Сервер представляет сочетание специализированного ПО и оборудования, предоставляемого службами в сети для остальных клиентских компьютеров (рабочих станций) или других процессов.

Архитектура "клиент-сервер" *(Client-server architecture)* по существу явилась первым вариантом распределённой сетевой архитектуры, т.е. двухуровневой распределённой системой. В приложениях "клиент–сервер" часть вычислительных операций и бизнес-логики перенесена на сторону клиента.

Клиент – приложение, посылающее запрос к серверу. Он отвечает за обработку, вывод информации и передачу запросов серверу. В качестве ЭВМ клиента может быть использован любой компьютер.

Сервер – персональная или виртуальная ЭВМ, выполняющая функции обслуживания клиента и распределяющая ресурсы системы: принтеры, базы данных, программы, внешнюю память и др.

Сетевой сервер поддерживает выполнение функций сетевой ОС, терминальный – выполнение функций многопользовательской системы.

Серверный процесс в архитектуре клиент-сервер – это процедура выполнения на сервере запроса клиентского процесса и отсылка ему ответа.

Администрирование в корпоративных сетях

Средства сетевого и системного администрирования не занимали доминирующих позиций в корпоративных информационных системах. Традиционно им отводилась вспомогательная роль. Это привело к тому, что структура и функции ПО данного класса оказались в прямой зависимости от архитектуры вычислительных систем и эволюционировали вместе с ними.

При организации сети компании, распределённой средствами Интернета по большой площади, нужно заботиться о надёжной маршрутизации, своевременном обмене информацией и о защите этой информации от несанкционированного доступа. Кроме того, следует организовать информационное обслуживание, единое для всех частей такой распределённой структуры. При построении корпоративных информационных сетей, как правило, используют две базовые архитектуры: Клиент-сервер и Интернет/Интранет. Одной из самых распространённых архитектур построения корпоративных информационных систем является архитектура “клиент-сервер”. В реализованной по данной архитектуре информационной сети клиенту предоставлен широкий спектр приложений и инструментов разработки, ориентированных на максимальное использование вычислительных возможностей клиентских рабочих мест. При этом ресурсы сервера в основном используются для хранения и обмена документами, а также с целью выхода во внешнюю среду. Для программных систем, имеющих разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, предоставляя возможность приложениям непосредственно адресоваться к другим серверным приложениям, или маршрутизировать запросы к ним. При этом частые обращения клиента к серверу снижают производительность работы сети. Кроме того, приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределённый характер построения системы обуславливает сложность её настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре “клиент-сервер”, тем выше вероятность отказа любого из её компонентов.

В основе архитектуры Интернет/Интранет корпоративных информационных систем лежит принцип “открытой архитектуры”, определяющий независимость реализации корпоративной системы от конкретного производителя.

Открытая архитектура (*Open architecture)* – архитектура компьютера или периферийного устройства, имеющая опубликованные спецификации, позволяющие другим производителям разрабатывать дополнительные устройства к системам с такой архитектурой. При этом важное место в любой многопользовательской среде отводится сетевым операционным системам.

Различные сетевые операционные системы и особенности администрирования в них

Сетевые системы обычно делят на: сетевые ОС, распределённые ОС и ОС мультипроцессорных ЭВМ.

Сетевые ОС – такие, в которых машины обладают высокой степенью автономности и общесистемных требований мало. Можно вести диалог с другой ЭВМ, вводить задания в её очередь пакетных заданий, иметь доступ к удалённым файлам, хотя иерархия директорий может быть разной для разных клиентов.

Распределённые ОС образуют единый глобальный межпроцессный коммуникационный механизм, глобальную схему контроля доступа, одинаковое видение файловой системы.

ОС мультипроцессорных ЭВМ – это единая очередь процессов, ожидающих выполнения, и одна файловая система.

Таблица 1.

Свойства сетевых ОС

Свойства компьютерной системы	Сетевая ОС	Распределенная ОС	ОС мультипроцессора
Выглядит как виртуальная однопроцессорная ЭВМ	НЕТ	ДА	ДА
Одна и та же ОС выполняется на всех процессорах	НЕТ	ДА	ДА
Количество копий ОС в памяти	N	N	1
Осуществление коммуникации	Разделяемые файлы	Сообщения	Разделяемая память
Наличие согласованного сетевого протокола	ДА	ДА	НЕТ
Наличие единой очереди выполняющихся процессов	НЕТ	НЕТ	ДА
Наличие хорошо определенной семантики разделения файлов	Обычно НЕТ	ДА	ДА

Важнейшая функция ОС – распределение ресурсов. В многозадачных (мультипрограммных) ОС приложения конкурируют между собой за ресурсы. От их распределения, зависит производительность всей вычислительной системы. Любые, особенно распределённые информационные ресурсы, требуют периодической реорганизации. Реорганизация данных является трудоёмкой операцией и при больших размерах БД может занять значительное время. При этом нельзя реорганизовать “текущую” БД. Сначала все пользователи должны её закрыть. Если для БД установлено разграничение прав доступа пользователей, то

её реорганизацию может выполнить только администратор БД. Перед выполнением операции система запросит имя и пароль администратора.

Мультипрограммирование *(Multitasking)* – способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются несколько программ. Способ предназначен для повышения эффективности использования вычислительной системы. В мультипрограммных системах распределением ресурсов между программами занимается подсистема управления процессами и потоками.

Поддержка работоспособности системы – основная задача её администрирования, поэтому администрирование сети осуществляется с помощью сетевых операционных систем. Рассмотрим их.

NetWare

Операционная система *NetWare* фирмы *Novell* работает на сервере и обеспечивает средства для рабочих станций. Основными функциями, обеспечиваемыми *NetWare*-сервером, являются управление файловой системой и планирование обработки задач. Сетевые средства представляют выполняемые на сервере приложения, основанные обычно на архитектуре “клиент-сервер”.

Протокол ядра *NetWare NCP (NetWare Core Protocol)* определяет служебные средства, доступные для пользователей этих сетей. Он прозрачен для пользователей и автономных приложений на рабочих станциях. Одно из наиболее важных средств *NetWare* – поддержка других ОС. При этом можно подключать рабочие станции, на которых работают *DOS, Windows, OS*/2 и *Unix*. Поддержка рабочих станций, *Windows* и *OS*/2 встроена в *NetWare*, а некоторые *сервисные* управляющие утилиты используют интерфейс *Window*s.

NetWare использует независимую от протокола структуру, известную как *ODI (Open Data-Link Interface),* обеспечивающую одновременную поддержку различных сетевых протоколов. Допускается использование различных сетевых плат. Пакеты направляются в соответствующий стек протокола над уровнем *ODI*, например, *IPX, TCP/IP*. На верхнем уровне протоколы обеспечивают поддержку файловой системы и различных ОС, устанавливаемых на *NetWare*-сервере. Аналогичная схема используется на рабочих станциях. Чтобы пользователи могли подключаться к сетям, применяющим различные коммуникационные протоколы, например, протокол *Unix TCP/IP*. *NetWare* предусматривает встроенные средства межсетевой маршрутизации, позволяющие объединять столько сетевых сегментов (*Token Ring, Ethernet, ArcNet* и др.), сколько сетевых плат будет содержать сервер. Связанные вместе сети представляются пользователям как одна сеть.

Windows

Существует несколько альтернативных возможностей использования ОС *Windows*, разработанной корпорацией *Microsoft*. В основном, они ориентирован на облегчение задач сетевого администрирования и установки *Windows* для пользователей, желающих обращаться к сети через ОС *Windows*. Фирма *Microsoft* разработала ОС *Windows*, ориентированную на многопользовательскую работу.

Чтобы подчеркнуть её принципиальную новизну, в название добавили символы *NT* (*New Technology* – новая технология). Её промышленный выпуск начался в 1993 году. Это была 32-разрядная ОС со встроенной сетевой поддержкой и развитыми многопользовательскими средствами. *Windows NT* обеспечивает: многозадачность, многопроцессорную работу, переносимость на различные платформы, защиту от несанкционированного доступа, заданный уровень секретности. Описываемые процедуры предусматривают копирование всех файлов *Windows* в совместно используемый каталог сети. Затем пользователи или супервизоры могут установить *Windows* на рабочих станциях, обращаясь к программам установки и файлам этого совместно используемого каталога, а не инсталлируя *Windows* непосредственна на рабочих станциях. Известно несколько методов работы с ОС *Windows*.

Метод 1. Пользователи полностью устанавливают *Windows* на своих рабочих станциях, обращаясь к программам установки и файлам в совместно используемом каталоге *Windows* на сервере. После этого *Windows* запускается с локального жёсткого диска.

Метод 2. На рабочую станцию копируются только персональные файлы конкретного пользователя Другие файл остаются в сети в каталоге, где они используются совместно с другими *пользователями*. *Windows* загружается из сети, но считывает и записывает файлы конфигурации конкретного пользователя.

Метод 3. Пользователи полностью запускают *Windows* из сети. Их личные файлы конфигурации хранятся в персональных каталогах, а совместно используемые файлы – в разделяемом каталоге, с которым могут работать все другие пользователи Windows.

Первый метод иногда считают наилучшим. При этом на жёстком диске рабочей станции должно быть достаточно места для размещения всех файлов *Windows*. Другой недостаток этого метода заключается то, что пользователь сам отвечает за обновление файлов ОС и приложений в его системе. Когда Windows устанавливается на сервере, эти задачи могут выполнять администраторы сети.

Второй метод позволяет сэкономить пространство на диске рабочей станции, так как там записывается всего несколько файлов ОС, таких как файлы *INI*. Однако при доступе пользователей к совместно используемым файлам ОС увеличивается сетевой трафик. Обновления выполняются на сервере, что облегчает задачи управления.

Третий метод самый простой с точки зрения администратора сети, но он создаёт наиболее интенсивный трафик, а потому обычно используется для запуска *Windows* с бездисковых рабочих станций.

Версия ОС Windows NT 4.0 выпускалась до 2000 года. Ей на смену, вышлаерсия5.0 под названием *Windows* 2000, в основе которой заложена технология *NT*. Windows 2000, имеет четыре модификации:

- *Professional* для рабочих станций (поддерживает двухпроцессорную ПЭВМ);

- *Server* для серверов малых локальных сетей (для четырёхпроцессорной ПЭВМ);
- *Advanced Server* для серверов больших локальных и удалённых сетей (до 16 процессоров);
- *Data Center Server* для крупных узлов сетей (поддерживает ЭВМ на 64 процессорах).

Затем в 2001 г. появляется настольная версия *Windows XP*, а в 2003 г. – серверная ОС – *Windows Server* 2003. В системах семейства этой серверной ОС немного принципиально новых решений, и они представляют эволюционное развитие серверных продуктов *Windows* 2000. Это более законченные и надёжные реализаци революционных, по сравнению с *Windows NT* 4.0, изменений, появившихся в *Windows* 2000. При этом семейство *Windows Server* 2003 унаследовало ряд возможностей системы *Windows XP*, отсутствовавших в *Windows* 2000. Четыре редакции ОС, образуют семейство *Windows Server* 2003 (*Standard, Enterprise, Datacenter и Web Edition*), которые в первую очередь различаются по степени масштабируемости и производительности.

Windows Server 2003, *Standard Edition* – универсальная сетевая система общего назначения, предназначенная для корпоративного использования небольшим компаниям или подразделениям крупных фирм при решении различных задач: поддержка служб печати и файловых сервисов, маршрутизация и удалённый доступ, обеспечение работы СУБД и т. д.

Windows Serve 2003, *Enterprise Edition* – платформа для развертывания бизнес-задач любого масштаба включая службы Интернета. При этом обеспечивается большая *производительнос и отказоустойчивость, чем при использовании Windows Server2003, Standard Edition*, достигаемые за счёт большего числа поддерживаемых процессоров, кластеризации и увеличенного объёма памяти.

Windows Server 2003, *Datacenter Edition* самая мощная из всех редакций *Windows Server* 2003. Она ориентирована на обеспечение максимального уровня производительности и надёжности для критически важных приложений и задач. В ней отсутствует ряд служб, целесообразных для использования в *небольших компаниях или группах.*

Windows Server 2003, Web Edition – новый продукт в семействе серверов *Microsoft*, в первую очередь предназначенный для веб-хостинга и поддержки *XML* веб-служб в небольших организациях и подразделениях.

UNIX

Одна из самых популярных в мире операционных систем – *UNIX* – разработана в конце 1960-х годов фирмой *Bell Laboratories AT&T*. Её сопровождает и распространяет большое число компаний.

Первоначально в середине 1970-х годов эта ОС создавалась как интерактивная многозадачная система для терминальной работы миникомпьютеров и мэйнфреймов. С тех пор она выросла в одну из наиболее распространённых ОС, несмотря на свой неудобный интерфейс и отсутствие централизованной стандартизации. До 1980 года *UNIX* использовалась в университетах и

правительственных исследовательских центрах. Основанная на наборе простых, но мощных инструментальных средств, эта ОС стала использоваться для разработки программных средств и получила промышленное применение. Первая коммерческая версия систе*мы* под названием *Xenix* выпущена в середине 1970-х годов фирмой *Microsoft*. Широкому её распространению способствовала бесплатная поставка в форме исходных текстов. Существенная особенность *UNIX* – переносимость на различные ЭВМ, так как её сетевая файловая система, лучше других ОС приспособлена для работы в сетях разнообразных компьютеров.

Семейство ОС *UNIX* в основном ориентировано на большие локальные и глобальные сети ЭВМ. ОС *UNIX* одновременно является операционной средой использования существующих прикладных программ и средой разработки новых приложений. Стандартным языком программирования в данной среде является язык Си (Си++). Это объясняется тем, что, во-первых, ОС UNIX написана на языке Си, а, во-вторых, язык Си является одним из наиболее качественно стандартизованных языков.

В ОС UNIX, как и в любой другой многопользовательской ОС, обеспечивающей защиту пользователей друг от друга и защиту системных данных от любого непривилегированного пользователя, имеется защищённое ядро, управляющее ресурсами компьютера и предоставляющее пользователям базовый набор услуг. Это не очень чётко структурированный монолит большого размера, поэтому программирование на уровне ядра ОС *UNIX* продолжает оставаться искусством.

Система обладает свойством высокой мобильности – вся ОС, включая её ядро, сравнительно просто переносится на различные аппаратные платформы. Все части системы, не считая ядра, являются полностью машинно-независимыми. Эти компоненты аккуратно написаны на языке Си, и их перенос на новую платформу обычно требует только перекомпиляция исходных текстов в коды целевого компьютера. Небольшая часть ядра машинно-зависимая. Она написана на смеси языков Си и Ассемблера целевого процессора. При переносе системы на новую платформу требуется переписать эту часть ядра с использованием языка Ассемблера и с учётом специфических черт целевой аппаратуры. Машинно-зависимые части ядра изолированы от основной машинно-независимой части. При хорошем понимании назначения каждого машинно-зависимого компонента переписывание машинно-зависимой части в основном является технической задачей, хотя и требует программистов высокой квалификации.

Средства общения с ядром в ОС *UNIX* называются системными вызовами. Для обращения к функциям ядра ОС используют “специальные команды” процессора, при выполнении которых возникает особое внутреннее прерывание процессора, переводящее его в режим ядра. В большинстве современных ОС этот вид прерываний называется “*trap*” – ловушка. При обработке таких прерываний (дешифрации) ядро ОС распознаёт, что данное прерывание является запросом к ядру со стороны пользовательской программы на выполнение определённых действий, выбирает параметры обращения и обрабатывает его, после чего выполняет “возврат из

прерывания", возобновляя нормальное выполнение пользовательской программы.

Поскольку ОС *UNIX* стремится обеспечить среду, в которой пользовательские программы полностью мобильны, потребовался дополнительный уровень, скрывающий особенности конкретного механизма возбуждения внутренних прерываний. Он обеспечивается "библиотекой системных вызовов" – обычной библиотекой с заранее реализованными функциями системы программирования языка Си. Внутри любой функции конкретной библиотеки системных вызовов содержится код, являющийся специфичным для данной аппаратной платформы. Каждому возможному прерыванию процессора соответствует фиксированный адрес физической оперативной памяти. Когда процессору разрешается прерваться из-за наличия внутренней или внешней заявки на прерывание, происходит аппаратная передача управления на ячейку физической оперативной памяти с соответствующим адресом. Обычно адрес этой ячейки называется "вектором прерывания". Как правило, заявки на внутреннее прерывание (поступающие непосредственно от процессора) удовлетворяются немедленно. ОС должна разместить в соответствующих ячейках оперативной памяти программный код, обеспечивающий начальную обработку прерывания и инициирующий полную обработку.

ОС *UNIX* требуется общая основа организации сетевых средств, основанных на многоуровневых протоколах. Для решения этой проблемы реализовано несколько механизмов, обладающих примерно одинаковыми возможностями, но не совместимых между собой, поскольку каждый из них являлся результатом индивидуального проекта. Слабая развитость в *UNIX* подсистемы ввода/вывода потребовала включения потоков, как механизма реализации существующего символьного ввода/вывода. Механизм потоков не навязывает конкретной архитектуры сети и (или) конкретных протоколов. Как любой другой драйвер устройства, потоковый драйвер представляет специальный файл файловой системы со стандартным набором операций: *open, close, read, write*. В *UNIX* протокол *TCP/IP* реализован как набор потоковых модулей плюс дополнительный компонент *TLI* (*Transport Level Interface* – интерфейс транспортного уровня). *TLI* является интерфейсом между прикладной программой и транспортным механизмом. Приложение, пользующееся интерфейсом *TLI*, получает возможность использовать *TCP/IP*. Интерфейс *TLI* основан на использовании классической семиуровневой модели *ISO/OSI.*

Перед началом работы зарегистрированный пользователь вводит со свободного терминала своё учётное имя (*account name*) и, возможно, пароль (*password*). Регистрацию новых пользователей выполняет администратор системы. Пользователь не может изменить своё учётное имя, но может установить и (или) изменить свой пароль. Пароли в закодированном виде хранятся в отдельном файле. Каждому зарегистрированному пользователю соответствует каталог файловой системы, называемый "домашним" (*home*) каталогом пользователя. При входе в систему пользователь получает неограниченный доступ к этому каталогу и всем содержащимся в нём каталогам и файлам. Он может создавать, удалять и

модифицировать каталоги и файлы в его домашнем каталоге. Потенциально он может получить доступ и к другим файлам. Однако такой доступ может быть ограничен, если пользователь не имеет достаточных привилегий.

Ядро ОС *UNIX* идентифицирует каждого пользователя по его идентификатору (*User Identifier, UID*) – уникальному целому значению, присваиваемому пользователю при регистрации в системе. Кроме того, каждый пользователь относится к некоторой группе пользователей, также идентифицируемой целым значением (*Group Identifier, GID*). Значения *UID* и *GID* для каждого зарегистрированного пользователя сохраняются в учётных файлах системы и приписываются процессу, в котором выполняется командный интерпретатор, запущенный при входе пользователя в систему. Эти значения наследуются каждым новым процессом, запущенным от имени данного пользователя, и используются ядром системы для контроля правомочности доступа к файлам, выполнения программ и т.д. Ограничения для пользователя касаются: максимального размера файла и числа сегментов разделяемой памяти, максимально допустимого пространства на диске и т.д.

Администратор системы также является зарегистрированным пользователем. Он обладает большими возможностями, чем обычные пользователи. В ОС *UNIX* ему выделяется одно нулевое значение *UID*. Пользователь с таким *UID* называется суперпользователем (*superuser*) или *root*. Он имеет неограниченные права доступа к любому файлу и на выполнение любой программы. Кроме того, такой пользователь может осуществлять полный контроль над системой; остановить и даже разрушить её. Супервизор должен хорошо знать базовые процедуры администрирования ОС *UNIX*. Он отвечает за безопасность системы, её правильное конфигурирование, добавление и исключение пользователей, регулярное копирование файлов и т.д. При этом на него не распространяются ограничения на используемые ресурсы.

После входа пользователя в систему для него запускается один из командных интерпретаторов. Общее название любого командного интерпретатора ОС *UNIX* – *shell* (оболочка), поскольку любой интерпретатор представляет внешнее окружение ядра системы. Оболочка – программа, создаваемая для упрощения работы со сложными программными системами. Оболочки преобразуют неудобный командный пользовательский интерфейс в дружественный графический интерфейс или интерфейс типа меню. Обычно оболочка реализуется в виде отдельной программы. Вызванный командный интерпретатор выдаёт приглашение на ввод пользователем командной строки, которая может содержать простую команду, конвейер команд или последовательность команд. Так будет до тех пор, пока пользователь не завершит сеанс работы путём ввода команды “*logout*” или нажатия комбинации клавиш “*Ctrl-d*”.

UNIX первая в истории мобильная ОС, обеспечивающая надёжную среду разработки и использования мобильных прикладных систем. Она представляет и практическую основу для построения открытых программно-аппаратных систем и комплексов. Для производства основанных на этой ОС совместимых ОС необходима стандартизация (интерфейсов) средств ОС на разных уровнях.

Одним из ранних стандартов де-факто ОС *UNIX* явился изданный *UNIX System Laboratories (USL)* документ *System V Interface Definition (SVID).* Это не единственный стандарт для ОС *UNIX*.

Контроль и управление сетью с *UNIX*-подобной ОС представляют сложную проблему, в решении которой выделяют два основных направления:

- сохранение административной управляемости;
- сохранение технической управляемости.

Административные проблемы обычно связывают с распределением сетевых ресурсов между различными подразделениями и пользователями, координацией их действий в процессе функционирования развития сети. Ключевым вопросом является способ хранения в системе *UNIX* указаний о владельце и привилегиях, связанных с файлом. Обычно, процесс, запущённый пользователем, имеет привилегии на доступ, принадлежащие этому пользователю. Однако есть системные команды доступа к файлам, к которым администраторы не хотят разрешать доступ пользователя. Администратор ведает всеми вопросами безопасности. Он должен вести постоянное наблюдение (в т. ч. упреждающее администрирование) за изменениями в системе и уметь противодействовать вмешательствам. Основная идея упреждающего администрирования сводится к тому, чтобы, проанализировав поведение АИС или отдельных её компонентов, предпринять превентивные меры, позволяющие не допустить развития событий по наихудшему сценарию. Системные администраторы должны проверять свои системы и смотреть на них с точки зрения нарушителя. Так, для предотвращения возможностей взлома системы, в первую очередь, нельзя оставлять без присмотра суперпользовательский терминал.

Создавать коммерческий *UNIX* большая и трудоёмкая работа. Часто для этого требуются сотни и даже тысячи программистов, специалистов по тестированию, писателей документации, административного персонала. В фирмах, разрабатывающих коммерческие *UNIX*, вся система создаётся под жёстким контролем качества. Существует система управления написанием программ, внесением изменений, документированием, информированием о выявленных ошибках и их устранением. Разработчикам запрещено по собственному желанию добавлять какие-либо свойства или менять критически важные коды. Они могут вносить изменения только, как реакцию на выявленные ошибки, документировать вносимые изменения так, чтобы можно было систему при необходимости “вернуть назад”. Каждый разработчик закреплён за одной или несколькими частями системного кода, и только он имеет право исправлять замеченные ошибки. Внутри фирм департаменты контроля качества осуществляют жёсткое тестирование любой новой версии ОС. Разработчики обязаны под контролем устранять выявленные ошибки. Существует сложная система статистического анализа, определяющая, сколько ошибок должно быть устранено, чтобы объявить переход к новой версии.

Linux

Linux (произносится “лИнукс”) – свободно распространяемая версия *UNIX*, разработана аспирантом Линусом Торвальдсом (*Linus Torvalds*) в Университете

Хельсинки (Финляндия) и впервые появилась в октябре 1991 года. Затем, во время сетевой конференции в 1992 году, он объявил, что в качестве “хобби” приступил к разработке *UNIX*-подобной компактной ОС для процессора I80386.

В рамках *UNIX*–систем была разработана ОС для ПЭВМ под названием *Linux*. Основное внимание в этой ОС уделялось созданию ядра. Вопросы поддержки работы с пользователем, документирования, тиражирования и т.п. обсуждались. Её особенность – открытый код. ОС поставляется в виде исходного текста, который можно модифицировать под конкретный состав и направление использования ЭВМ. *Linux* распространяется бесплатно и считается самой быстроразвивающейся ОС в области многопользовательских многозадачных систем. Это гибкая полноценная многозадачная многопользовательская ОС семейства *UNIX*-подобных ОС, способна работать с *X Windows, TCP/IP, Emacs* (редактор текста), *UUCP, mail* и *USENET*. При этом множество пользователей может одновременно работать на одной машине и выполнять много программ.

X Windows (Система *X Window* или кратко просто *X*) – стандартный графический интерфейс для *UNIX*-машин, благодаря которому пользователь может одновременно видеть на экране компьютера несколько окон, при этом каждое окно имеет независимый *login*.

UUCP (UNIX-to-UNIX Copy) – старейший механизм передачи файлов, электронной почты и электронных новостей между *UNIX*-машинами. Классически *UUCP*-машины связываются друг с другом по телефонным линиям через модем, но *UUCP* может использовать в качестве транспортного средства и связь по *TCP/IP*.

Практически все важнейшие современные программные пакеты используются под *Linux*. Это уникальная операционная система. Чтобы эффективно её использовать, важно понимать её философию и особенности проектирования. Это большая и достаточно сложная система для решения сложных задач и организации распределённых вычислений – отличный выбор для персональных вычислений в среде *UNIX*.

Нет определённой организации, отвечающей за развитие данной системы. *Linux* функционирует и развивается “на общественных началах” *(free implementation*) группой добровольцев, первоначально в кругу пользователей Интернета, обменивавшихся кодами, информацией об обнаруженных ошибках, выявлявших проблемы, возникавшие при расширении сферы применения. Большинство свободно распространяемых в Интернете программ для *UNIX* может быть откомпилировано для *LINUX* практически без особых изменений. Большей частью *Linux*-сообщество общается через группы по интересам *USENET*. Любому, желающему включить свой код в “официальное” ядро программы, следует написать об этом письмо Линусу Торвальдсу (создателю *Linux*), проводящему тестирование. Если предлагаемый код вписывается в систему и не противоречит её принципам, скорее всего он будет включен в ядро. При этом все исходные тексты для этой ОС, включая ядро, драйверы устройств, библиотеки, пользовательские программы и инструментальные средства распространяются свободно. Применительно к *Linux* можно не учитывать *UNIX* концепцию организации

разработки большой программной системы, отладки, контроля качества, статистического анализа и т.п.

Linux проста в инсталляции и использовании. Она обеспечивает полный набор протоколов *TCP/IP* для сетевой работы и услуг *TCP/IP (FTP, telnet, NNTP и SMTP)*. Ядро Linux поддерживает загрузку только нужных страниц, то есть с диска в память загружаются те сегменты программы, которые действительно используются. При этом возможно использование одной страницы, физически один раз загруженной в память, несколькими выполняемыми программами. Для увеличения объёма доступной памяти *Linux* осуществляет разбиение диска на страницы: то есть на диске может быть выделено до 256 Мбайт "пространств для свопинга" (*swap space* – место обмена). Когда систем необходимо использовать больше физической памяти, она с помощью свопинга выводит неактивные страницы на диск, что позволяет выполнять более объёмные программы и обслуживать одновременно больше число пользователей. Свопинг не исключает наращивания физической памяти, поскольку он снижает быстродействие и увеличивает время доступа.

Выполняемые программы используют динамически связываемые библиотеки, т.е. они могут совместно использовать библиотечную программу, представленную одним физическим файлом на диске. Это позволяет выполняемым файлам занимать меньше места на диске, особенно тем, которые многократно используют библиотечные функции. Есть также статические связываемые библиотеки для тех, кто желает пользоваться отладкой на уровне объектных кодов или имеет "полные" выполняемые программы, которые не нуждаются в разделяемых библиотеках. В *Linux* разделяемые библиотеки динамически связываются во время выполнения, позволяя программисту заменять библиотечные модули своими собственными.

Linux идеален для создания *UNIX*-приложений. Он обеспечивает полную *UNIX*-среду программирования, включая все стандартные библиотеки, программный инструментарий, компиляторы, отладчики, которые встречаются и в других *UNIX*-системах. Профессиональные *UNIX*-программисты и системные администраторы могут использовать *Linux* на домашних компьютерах, а с них переносить написанные программы на компьютеры организации (фирмы). Такой метод позволяет экономить время и деньги, обеспечивает комфортабельную работу на домашнем компьютере. *Linux* прежде всего ориентирован на разработчиков. Однако любой человек, имеющий достаточные знания и навыки, может принять участие в совершенствовании и отладке ядра, переносе в *Linux* новых программ, написании документации, помощи новичкам.

Сама по себе система проектируется по открытому принципу. Число вносимых радикальных изменений в систему постепенно уменьшается. Однако общая тенденция заключается в выдаче новой версии ядра. Новые версии ядра появляются примерно раз в несколько недель. Постоянно появляются новые программы. Учитывая такую динамику большинству пользователей лучше делать частичные усовершенствования, то есть менять те части ОС, которые действительно нуждаются в обновлении.

В *Linux* много типов оболочек, позволяющих настраивать систему под личные нужды пользователей. Она содержит динамические библиотеки (*DLL*), позволяющие экономить место, поскольку они вызываются только во время выполнения. Эти библиотеки позволяют прикладному программисту переопределять функции, включая свои коды. Инсталляция и использование личного ПО *Linux* не требует большой подготовки. Но при решении более сложных задач эксплуатации *Linux* – инсталляция новых программ, перекомпиляция ядра и т.п. – необходимы базовые знания *UNIX*. При этом, эксплуатируя *Linux*, можно освоить все существенные особенности *UNIX*, необходимые для выполнения этих задач. Студенты, изучающие ВТ и программирование, могут использовать *Linux* для обучения программированию в *UNIX* и изучения архитектуры ядра. Через *Linux* можно получить доступ к полному набору библиотек и утилит, а также к исходным текстам ядра и библиотек.

Сравнение сетевых ОС

Поскольку *UNIX* – мощная операционная среда с хорошим управлением памятью, большой доступной вычислительной мощностью и наиболее распространена в сетях общего пользования, то проведём с ней сравнение рассмотренных сетевых ОС.

NetWare и *Unix* – это совершенно разные сетевые ОС. Некоторые работающие в *NetWare* программы обработки документов взаимодействуют либо с модулями на *Unix*-системах или с базами данных *SQL*. Эта связь чётко отражает идею выбора *Unix* в качестве используемого механизма базы данных и применения *NetWare* как средства представления информации пользователю. При работе с *Unix* используется стык протоколов *TCP/IP*, а при работе с *NetWare* – *IPX/SPX*. Вообще, протоколы *IPX/SPX* и *TCP/IP* можно загрузить на любой существующей компьютерной платформе. При необходимости можно смешать эти протоколы. Во многих случаях комбинирования *NetWare и Unix* сводится к задаче трансляции. Транслятор для *Unix* и *NetWare* – это сетевой интерфейс *TPC/IP* (“шлюз”).

OS/2 и *Windows NT*, как и *Linux*, являются полными многозадачными ОС. *OS*/2 и *Windows NT* принадлежат частным компаниям. Поэтому, интерфейс и проектные решения контролируются конкретными фирмами, и только они могут совершенствовать свои продукты. Такая организация дела имеет и преимущества: обеспечивается жёсткая стандартизация программного и пользовательского интерфейсов.

Мощные *UNIX* имеют более высокую степень масштабируемости, чем *Windows NT*/2000. В отличие от *UNIX* и *Linux*, в *Windows NT* отсутствует система квотирования дискового пространства, обеспечивающая защиту от переполнения дисков и гарантирующая справедливое распределение пространства между пользователями.

При сравнении *UNIX* и *Linux* важно понимать различия между *Linux* и другими реализациями *UNIX* для персональных компьютеров, а также с другими ОС (*Windows NT, OS/2* и др.). Прежде всего, *Linux* на одной и той же машине может сосуществовать с другими ОС. Почти все коммерческие версии *UNIX*

поддерживают практически одинаковую программную среду и сетевые характеристики. Однако имеются и значительные отличия между *Linux* и коммерческими ОС *UNIX*. Прежде всего, *Linux* поддерживает иной спектр аппаратных средств – обычно это хорошо известные устройства, большую часть которых реально имеют пользователи.

Коммерческие реализации *UNIX* обычно обеспечены полным набором документации, а также обязательствам разработчика по сопровождению. Документация на *Linux* ограничивается материалами, присутствующими в Интернете и отельными книжками. *Linux* не менее надёжен, чем коммерческие версии *UNIX*. Она распространяется свободно через Интернет и её можно купить на дисках *(CD* или *DVD-ROM*), скопировать у кого-нибудь или разделить с ним стоимость покупки ОС. Для инсталляции *Linux* на большом количестве машин достаточно купить одну копию, а на тиражирование нет лицензионных ограничений. *Linux* требует мало памяти в сравнении с другими развитыми ОС, хотя, чем больше памяти, тем быстрее работает система. Большинство пользователей *Linux* выделяют часть жёсткого диска для области свопинга, используемой как виртуальная *RAM*. Хотя область свопинга не заменяет действительной физической памяти, она позволяет выполнять более объёмные приложения, удаляя неактивную часть программы на диск. При этом в *Linux* недостаточно хорошо реализована поддержка многопроцессорных конфигураций.

Технически, *OS*/2, *Windows NT* и *Linux* очень похожи: они имеют похожие интерфейсы с пользователем, систему защиты и т.п. Главное отличие состоит в том, что *Linux* есть разновидность *UNIX*, а отсюда все преимущества принадлежности к *UNIX*-сообществу. *UNIX* не только самая популярная ОС для многопользовательских машин, но и база для большей части свободно распространяемых в мире программ. Почти все программы, свободно доступные в Интернете, написаны для *UNIX*, даже глобальная сеть в большой степени построена с использованием *UNIX*. Интерфейс *UNIX* постоянно совершенствуется и меняется. Несколько организаций пытаются выработать стандарт программной модели, но эта очень сложная задача. *UNIX* дорогое ПО. При этом наряду со стоимостью собственно программ *UNIX*, в стоимость входят стоимость документации, сопровождения и гарантия качества. Это очень важные составляющие для больших организаций, но не столь существенные для индивидуальных пользователей.

Сложно однозначно говорить о том, какая из рассмотренных систем лучше. Для одних задач и в определённых случаях лучше подходят *UNIX* и *Linux*, в других ситуациях – *Windows NT* и др. При этом разнообразные сети различным образом взаимодействуют между собой и всё более очевидна необходимость их конвергенции (*Fixed Mobile Convergence, FMC*). Конвергенция (*Convergence*) информационных технологий – процесс сближения разнородных электронных технологий в результате их быстрого развития и взаимодействия. Сетевая конвергенция на основе *FMC* означает формирование единой инфраструктуры, предназначенной для предоставления клиентам фиксированных, мобильных и конвергентных сервисов. Разработчики считают, что будущее не за технологиями, а за услугами связи – абоненту важны не способ и средства доставки, а

единообразный и качественный сервис, предоставляемый ему в независимости от места нахождения, типа используемой сети и клиентского терминала.

Администрирование в сетях с операционными системами типа Windows

Для работы в локальной сети на сервер обычно устанавливают ОС *Windows NT Server*, а на рабочие узлы сети – версию *Windows NT – WorkStation*. Как правило, большинство системных инструментов требует наличия у запускающего их пользователя административных привилегий. Чтобы иметь возможность выполнения операций по управлению системой, администратор должен зарегистрироваться в системе под учётной записью, обладающей соответствующими правами. Однако правила безопасности требуют от администратора постоянно не пользоваться в системеподобными учётными записями.

Команда *RunAs* позволяет администратору выполнять всю работу по управлению системой, зарегистрировавшись в ней с использованием учётной записи рядового пользователя с весьма ограниченными правами. С её помощью администратор запускает любые утилиты от имени “уполномоченного” пользователя. При этом можно задействовать учётную запись администратора, или учётную запись пользователя, обладающего необходимыми правами. Команду *RunAs* можно использовать для установки и проверки пользовательских разрешений на доступ к файлам или объектам *Active Directory*. Для задания пользователю разрешений необходимо запустить соответствующую утилиту с административными привилегиями. Одновременно можно запустить требуемое приложение в контексте полномочий рассматриваемого пользователя и проверить результирующие разрешения. Таким образом, задача будет решена и проверена без необходимости повторной регистрации в системе под разными учётными записями.

Создание учётных записей и групп занимает важное место в обеспечении безопасности *Windows Server* 2003. Назначая им права доступа и привилегии, администратор получает возможность ограничить пользователей в доступе к конфиденциальной информации в сети, разрешить или запретить им выполнение в сети определённого действия, например, архивацию данных или завершение работы компьютера. В системах *Windows XP* и *Windows Server* 2003 имеется стандартный механизм *Remote Desktop for Administration*, или просто *Remote Desktop*, позволяющий удалённо подключаться и выполнять необходимые операции по управлению сервером. Этот механизм использует службы терминалов и поддерживает два одновременных удалённых подключения (в *Windows XP* – одно). Администратор может с любого рабочего места администрировать все серверы, находящиеся под управлением *Windows Server* 2003, подключаясь к ним удалённо. Отделение механизма удалённого администрирования от служб

терминалов позволило свести к минимуму нагрузку на сервер в ситуации управления сервером с другого компьютера.

В системах *Windows XP* и *Windows Server* 2003 имеется функция *Remote Assistance*, позволяющая пользователю инициировать доступ к своему компьютеру и получать помощь в сложных ситуациях. Для её использования обе системы должны работать под управлением *Windows XP* или *Windows Server* 2003. При включении *Remote Assistance* по умолчанию разрешено и удалённое управление компьютером. При этом для управления компьютером всегда требуется дополнительное, явное разрешение со стороны "хозяина" этого компьютера во время сеанса работы удалённого помощника. Если на компьютере под управлением *Windows XP* или *Windows Server* 2003 установлен веб-сервер в составе служб *Internet Information Services (IIS),* то через этот компьютер можно осуществлять удалённый доступ к любой системе *Windows XP* или Windows Server 2003, находящейся в той же локальной сети, из web-браузера Internet Explorer, работающего в любой ОС. Такая возможность позволяет, например, на маломощном компьютере под управлением *Windows* 95 запустить браузер и, введя имя удалённой системы *Windows Server* 2003 на базе какого-нибудь мощного процессора, работать на ней в полноэкранном режиме. Все сессии удалённого доступа шифруются, чтобы исключить несанкционированный доступ к данным и системам. Для управления средой пользователя предназначены средства систем *Windows* 2000/*XP* и *Windows Server* 2003.

Профили пользователей. В профиле пользователя хранятся все, определённые им, настройки рабочей среды системы, например, настройки экрана и соединения с сетью. Они автоматически сохраняются в папке.

Сценарий входа в систему (сценарий регистрации) представляет командный файл с расширением "*bat*" или "*cmd*", исполняемый файл с расширением "*exe*" или сценарий "*VBScript*", запускающийся при каждой регистрации пользователя в системе или выходе из неё. Он может содержать команды ОС, предназначенные, например, для создания соединения с сетью или для запуска приложения. С его помощью можно устанавливать значения переменных среды, указывающих пути поиска, каталоги для временных файлов и другую подобную информацию. Сервер сценариев *Windows* предназначен для выполнения сценариев прямо на рабочем столе Windows или в окне консоли команд. При этом сценарии ненужно встраивать в документ *HTML*. Сервер сценариев позволяет применять в ОС *Windows* простые мощные и гибкие сценарии. Раньше единственным языком сценариев, поддерживаемым *ОС Windows, был язык команд MS-DOS* (командный файл). Этот быстрый и компактный язык по сравнению с языками *VBScript* и *JScript*, обладает ограниченными возможностями. Архитектура сценариев *ActiveX* позволяет использовать все средства таких языков сценариев, как *VBScript* и *JScript*, одновременно сохраняя совместимость с набором команд *MS-DOS*. Для конфигурирования, поиска, выделения памяти определённым программам и управления приложениями ОС *Windows Server* 2003 и прикладные программы требуют определённой информации, называемой переменными среды (*environment*

variables) системы и пользователя. Эти переменные похожи на устанавливаемые в ОС *MS-DOS*, например, *PATH* и *TEMP*.

Системные переменные среды определяются в *Windows Server* 2003 независимо от того, кто зарегистрировался на компьютере. Если зарегистрировался член группы *Administrators*, то он может добавить новые переменные или изменить их значения. Переменные среды пользователя устанавливаются индивидуально для каждого пользователя одного и того же компьютера. После изменения переменных среды их новые величины сохранятся в реестре. Полный путь в командной строке команды *path* создаётся присоединением путей, устанавливаемых в файле *Autoexec.bat*, к путям, определённым в окне *Environment Variables*.

Аудит – процесс фиксирования событий, происходящих в ОС и имеющих отношение к безопасности: например, регистрация в системе или попытки создания объекта файловой системы, получения к нему доступа или удаления. Информация о подобных событиях заносится в файл журнала событий ОС. Полученную в результате информацию (журнал *Security*) можно просмотреть с помощью *Event Viewer* (Просмотр событий). В процессе настройки аудита указывают, какие события должны быть отслежены. Информация о них помещается в журнал событий. Каждая запись журнала хранит данные о типе выполненного действия, пользователе, выполнившем его, о дате и моменте времени выполнения данного действия. Аудит позволяет отслеживать любые попытки выполнения определённого действия. В журнале событий отражаются все попытки выполнения, в том числе неразрешённых действий. Для настройки аудита необходимо иметь права администратора.

С помощью *планировщика заданий* (*Task Scheduler*) можно составить расписание запуска командных файлов, документов, обычных приложений или различных утилит для обслуживания системы. Программы могут запускаться однократно, ежедневно, еженедельно или ежемесячно в заданные дни, при загрузке системы или регистрации в ней, а также при бездействии системы (*idle state*). Планировщик позволяет задавать сложное расписание для выполнения заданий, включающее продолжительность задания, время его окончания, количество повторов, зависимость от состояния источника питания (работа от сети или от батарей) и т. п. Задание сохраняется как файл с расширением "*job*", который можно перемещать с одного компьютера на другой. Администраторы могут создавать файлы заданий для обслуживания систем и переносить их в нужное место. К папке заданий можно обращаться удалённо, кроме того, задания можно пересылать по электронной почте. При перемещении файла "*.j*ob*" в другую систему необходимо восстановить разрешения на его использование, поскольку эти полномочия хранятся в системе безопасности *Windows*. При создании задания указывают имя и пароль пользователя, определяющие контекст безопасности, в котором выполняется задание. Это позволяет запускать на одном компьютере несколько заданий с различными правами в отношении безопасности, т. е. несколько пользователей могут одновременно иметь индивидуальные, независимые расписания запланированных заданий.

Администрирование в среде *Unix/Linux*

После того как физическая сеть собрана, администратор должен собственноручно назначить на каждой машине адреса интерфейсам. Обычно это делается с консоли того компьютера, который настраивается для работы в сети. Существует возможность динамической настройки с одного рабочего места всех машин сети. Однако, кроме явных преимуществ, у такого подхода есть срытые недостатки. Главный из них – это учёт статистики работы с каждой из машин системы. При динамическом назначении адресов машина может в разное время получать разные адреса, что не позволяет по адресу проидентифицировать машину. Многие же системы анализа трафика основываются на том, что соответствие между адресом и компьютером неизменно. Именно на этом принципе построены многие системы защиты от несанкционированного доступа.

Другой причиной, заставляющей жёстко назначать адреса компьютерам сети, является необходимость организации информационных сервисов на серверах сети. *TCP/IP* не имеет механизма оповещения рабочих мест о месте нахождения сервиса. Широковещание вообще не очень распространено в сетях *TCP/IP*, в отличие от сетей *Novell* или *Microsoft*. Каждый хост знает о наличии того или иного сервиса из файла своей настройки (например, указываются шлюз в другие сети или сервер доменных имен), или из файлов настроек прикладного ПО. Так, например, сервер *WWW* не посылает никакого широковещательного сообщения о том, что он установлен на данном компьютере в данной сети. Преимущество такого подхода заключается в низком трафике, порождаемом сетью *TCP/IP*. Этот трафик иногда значительно отличается, например, от трафика *Novell*. Кроме этого практически любое оборудование позволяет фильтровать трафик *TCP/IP*, что сильно облегчает сегментацию сети и делает её легко структурируемой. Для монтирования удалённой файловой системы нет необходимости использовать межсетевой протокол для доставки протоколов локальной сети, так как стек *TCP/IP* сам реализует этот межсетевой обмен.

В рамках организации сети *TCP/IP* уделяется внимание организации удалённых рабочих мест программистов и других сотрудников, использующих электронную связь для оперативного обмена информацией, доступа к информационным ресурсам, обмена электронной почтой и др. Организовать такие рабочие места в рамках сетей *TCP/IP* можно без особых проблем. Подключение локальной сети *TCP/IP* к Интернет осуществляется через местного провайдера. Обычно это та же организация, у которой был получен блок адресов для локальной сети.

Назначение и функционирование брандмауэра

При всех достоинствах сети *TCP/IP* имеют один врожденный недостаток – в них отсутствуют встроенные способы защиты информации от несанкционированного доступа, поскольку информация при способе доступа – удалённый терминал – передаётся по сети открыто. Это означает, что кто-то может найти способ просмотреть передаваемые по сети пакеты, а значит получить

коллекцию идентификаторов и паролей пользователей *TCP/IP* сети. Способов совершить подобные действия множество. Сходные проблемы возникают при организации доступа к архивам *FTP* и серверам *WWW*. Поэтому одним из основных принципов администрирования *TCP/IP* сетей является выработка общей политики безопасности: администратор определяет правила типа “кто, куда и откуда имеет право использовать те или иные информационные ресурсы”.

Эти проблемы в сетях обычно решаются путём установления специальных защитных программных и программно-технических средств – брандмауэров (*FireWall* – межсетевых фильтров, “стен”).

Управление безопасностью начинается с управления таблицей маршрутов. При статическом администрировании маршрутов включение и удаление последних производится вручную, в случае динамической маршрутизации эту работу выполняют программы поддержки динамической маршрутизации. Следующий этап – управление системой доменных имен, определение разрешений на копирование описания домена и контроля запросов на получение *IP*-адресов. Следующий барьер – системы фильтрации *TCP/IP* трафика. Наиболее распространённым средством такой борьбы являются системы *FireWall*. Используя эти программы, можно определить номер протокола и номер порта, по которым можно принимать пакеты с определённых адресов и отправлять пакеты на определённые адреса. Наконец, последнее средство защиты – шифрация трафика. Для этой цели используется различное программное обеспечение, разработанное для организации защищённого обмена через общественные сети.

5. Виртуальные частные сети

Частные сети используются организациями для соединения с удаленными сайтами и с другими организациями. Частные сети состоят из каналов связи, арендуемых у различных телефонных компаний и поставщиков услуг интернета. Эти каналы связи характеризуются тем, что они соединяют только два объекта, будучи отделенными от другого трафика, так как арендуемые каналы обеспечивают двустороннюю *связь* между двумя сайтами.

Частные сети обладают множеством преимуществ.

- Информация сохраняется в секрете.
- Удаленные сайты могут осуществлять обмен информацией незамедлительно.
- Удаленные пользователи не ощущают себя изолированными от системы, к которой они осуществляют доступ.

К сожалению, этот тип сетей обладает одним большим недостатком – высокой стоимостью. Использование частных сетей – очень дорогое удовольствие. Используя менее скоростные каналы связи, можно сэкономить деньги, но тогда удаленные пользователи начнут замечать недостаток в скорости, и некоторые из указанных выше преимуществ станут менее очевидными.

С увеличением числа пользователей интернета многие организации перешли на использование виртуальных частных сетей (*VPN*). *Виртуальные частные сети* обеспечивают многие преимущества частных сетей за меньшую цену. Тем не менее, с внедрением *VPN* появляется *целый* ряд вопросов и опасностей для организации. Правильно построенная виртуальная частная *сеть* может принести организации большую пользу. Если же *VPN* реализована некорректно, вся *информация*, передаваемая через *VPN*, может быть доступна из интернета.

Определение виртуальных частных сетей

Итак, мы намереваемся передавать через *интернет* секретные данные организации без использования арендуемых каналов связи, по-прежнему принимая все меры для обеспечения *конфиденциальности трафика*. Каким же образом нам удастся отделить свой трафик от трафика остальных пользователей глобальной сети? Ответом на этот вопрос является *шифрование*.

В интернете можно встретить трафик любого типа. Значительная часть этого трафика передается в открытом виде, и любой *пользователь*, наблюдающий за этим трафиком, сможет его распознать. Это относится к большей части почтового и веб-трафика, а также сеансам связи через протоколы telnet и *FTP*. Трафик *Secure Shell* (*SSH*) и *Hypertext Transfer Protocol Secure* (*HTTPS*) является шифруемым трафиком, и его не сможет просмотреть *пользователь*, отслеживающий пакеты. Тем не менее, трафик типа *SSH* и *HTTPS* не образует виртуальную частную *сеть VPN*.

Виртуальные частные сети обладают несколькими характеристиками.

- Трафик шифруется для обеспечения защиты от прослушивания.
- Осуществляется аутентификация удаленного сайта.
- Виртуальные частные сети обеспечивают поддержку множества протоколов.
- Соединение обеспечивает связь только между двумя конкретными абонентами.

Так как *SSH* и *HTTPS* не способны поддерживать несколько протоколов, то же самое относится и к реальным виртуальным частным сетям. *VPN*-пакеты смешиваются с потоком обычного трафика в интернете и существуют отдельно по той причине, что данный трафик может считываться только конечными точками соединения.

Рассмотрим более детально каждую из характеристик *VPN*. Выше уже говорилось о том, что трафик *VPN* шифруется для защиты от прослушивания. *Шифрование* должно быть достаточно мощным, чтобы можно было гарантировать *конфиденциальность* передаваемой информации на тот период, пока она будет актуальна. Пароли имеют срок действия, равный 30 дням (подразумевается политика изменения пароля через каждые 30 дней); однако секретная *информация* может не утрачивать своей ценности на протяжении долгих лет. Следовательно, *алгоритм* шифрования и применение *VPN* должны предотвратить нелегальное *дешифрование* трафика на несколько лет.

Вторая характеристика заключается в том, что осуществляется *аутентификация* удаленного сайта. Эта характеристика может требовать аутентификацию некоторых пользователей на центральном сервере либо взаимную аутентификацию обоих узлов, которые соединяет *VPN*. Используемый механизм аутентификации контролируется политикой. Политика может предусмотреть аутентификацию пользователей по двум параметрам или с использованием динамических паролей. При *взаимной аутентификации* может потребоваться, чтобы оба сайта демонстрировали *знание* определенного общего секрета (под

секретом подразумевается некоторая *информация*, заранее известная обоим сайтам), либо могут потребоваться *цифровые сертификаты*.

Виртуальные частные сети обеспечивают поддержку различных протоколов, в особенности на прикладном уровне. Например, удаленный *пользователь* может использовать протокол *SMTP* для связи с почтовым сервером, одновременно используя *NetBIOS* для соединения с файловым сервером. Оба указанных протокола могут работать через один и тот же цикл связи или канал *VPN* (см. рис. 1).

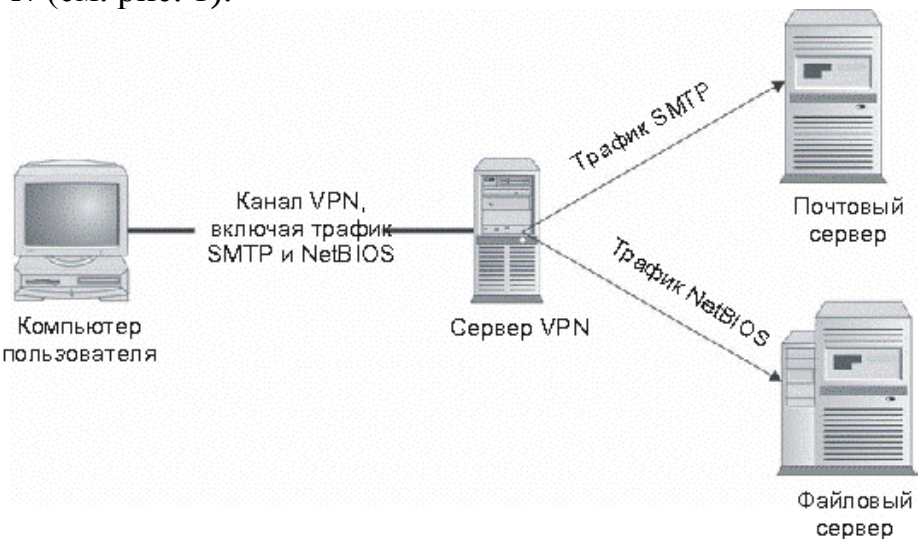


Рис. 1. Виртуальные частные сети поддерживают множество протоколов

VPN соединяет два конкретных объекта, образуя таким образом уникальный *канал связи* между двумя абонентами. Каждая из конечных точек *VPN* может единовременно поддерживать несколько соединений *VPN* с другими конечными точками, однако каждая из точек является отдельной от других, и трафик разделяется посредством шифрования.

Виртуальные частные сети, как правило, подразделяются на два типа: пользовательские *VPN* и узловые *VPN*. Различие между ними заключается в методе использования, а не в способе отделения трафика каждым из двух типов сетей. В оставшейся части данной лекции будет детально рассказываться о каждом из типов *VPN*.

Развертывание пользовательских виртуальных частных сетей

Пользовательские *VPN* представляют собой *виртуальные частные сети*, построенные между отдельной пользовательской системой и узлом или сетью организации. Часто пользовательские *VPN* используются сотрудниками, находящимися в командировке или работающими из дома. *Сервер VPN* может являться межсетевым экраном организации либо быть отдельным *VPN*-сервером. *Пользователь* подключается к интернету через телефонное подключение к локальному поставщику услуг, через канал *DSL* или *кабельный модем* и инициирует *VPN*-соединение с узлом организации через *интернет*.

Узел организации запрашивает у пользователя *аутентификационные данные* и, в случае успешной аутентификации, позволяет пользователю осуществить *доступ* ко внутренней сети организации, как если бы *пользователь* находился внутри узла и физически располагался внутри сети. Очевиден тот факт, что скорость сетевого соединения будет ограничиваться скоростью подключения пользователя к интернету.

Пользовательские *VPN* позволяют организациям ограничивать *доступ* удаленных пользователей к системам или файлам. Это ограничение должно базироваться на политике организации и зависит от возможностей продукта *VPN*.

В то время как *пользователь* имеет *VPN*-соединение с внутренней сетью организации, он также может соединяться и работать с интернетом или выполнять другие действия как обычный *пользователь* интернета. *Сеть VPN* поддерживается отдельным приложением на компьютере пользователя (см. рис. 2).

Рис. 2. Конфигурация пользовательской *VPN*

Преимущества пользовательских *VPN*

Пользовательские *VPN* обладают двумя основными преимуществами:

- Сотрудники, находящиеся в командировке, могут осуществлять доступ к электронной почте, файлам и внутренним системам в любое время без необходимости в осуществлении дорогостоящих междугородних и международных телефонных вызовов для соединения с серверами.
- Сотрудники, работающие из дома, могут осуществлять доступ к службам сети, как и сотрудники, работающие в организации, без аренды дорогостоящих выделенных каналов.

Оба эти преимущества можно приписать к экономии денежных средств. Экономия может заключаться в отказе от использования дорогостоящих междугородних и международных соединений, арендуемых каналов связи или в выполнении сотрудниками задач по администрированию серверов, принимающих входящие телефонные соединения. Домашние пользователи с *DSL* или кабельными

модемами могут добиться увеличения скорости при использовании линий телефонной связи со скоростями 56 Кбит/с. Все больше гостиничных номеров оборудуются соединениями для доступа в *сеть*, поэтому для пользователей, находящихся в поездке, создаются все условия для высокоскоростного доступа в сеть.

Проблемы, связанные с пользовательскими *VPN*

Правильное использование пользовательских *VPN* может снизить *затраты* организации, но пользовательские *VPN* не являются решением всех возможных проблем. При их использовании имеют *место* значительные риски, связанные с безопасностью, и проблемы реализации, с которыми приходится считаться.

Возможно, самой большой проблемой безопасности при использовании *VPN* сотрудником является одновременное соединение с другими сайтами интернета. Как правило, *программное обеспечение VPN* на компьютере пользователя определяет, должен ли трафик передаваться через *VPN*, либо его необходимо отправить на какой-либо другой *сайт* в открытом виде. Если на *компьютер* пользователя была произведена *атака* с использованием "*троянского коня*", возможно, что некий внешний нелегальный *пользователь* использует *компьютер* сотрудника для подключения к внутренней сети организации (см. рис. 3). Атаки данного типа осуществляются довольно сложно, но они совершенно реальны.

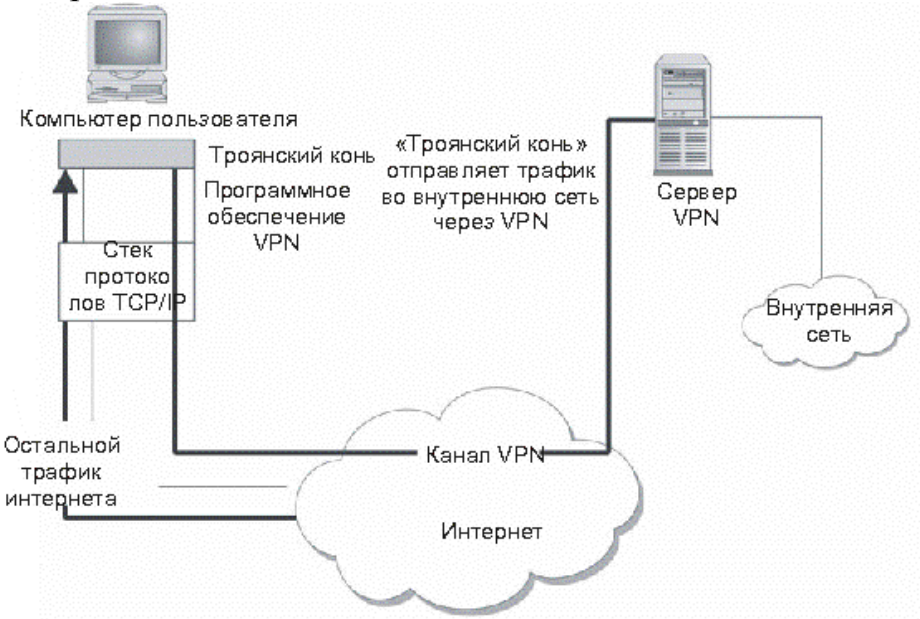


Рис. 3**.** Использование "троянского коня" для проникновения во внутреннюю сеть организации

Если управление *VPN*-пользователями не связано с центральной системой управления пользователями, этот факт должен учитываться в процедурах управления пользователями, покидающими организацию.

Пользователи должны проходить аутентификацию перед использованием сетей *VPN*. Так как *VPN* позволяет осуществлять удаленный *доступ* ко внутренней сети организации, эта *аутентификация* должна быть двухфакторной, то есть

запрашивать два аутентификационных параметра. Одним из параметров может являться сам *компьютер* пользователя. В этом случае вторым параметром должно быть нечто известное пользователю или непосредственно с ним связанное. В любом случае, второй *параметр* не должен находиться на компьютере и не должен быть с ним связан.

В организациях должна приниматься в расчет нагрузка трафиком. Главной точкой нагрузки является *VPN-сервер* в узле организации. *Ключевым параметром* нагрузки является ожидаемое число одновременных соединений. При установке каждого соединения *VPN-сервер* должен иметь возможность расшифровывать дополнительный трафик. Хотя *процессор* может обеспечивать поддержку больших объемов трафика, он может не обеспечивать *шифрование* и расшифровку большого числа пакетов без значительных задержек. Следовательно, *сервер VPN* должен создаваться с учетом ожидаемого числа единовременных соединений.

Еще один момент может повлиять на использование организацией пользовательской *VPN*. Он связан с использованием трансляции сетевых адресов (*NAT*) на противоположном конце соединения. Если ожидается, что сотрудники организации будут пытаться использовать *VPN* с узлов, защищенных межсетевыми экранами, могут возникнуть проблемы. Например, если организация А является консалтинговой компанией с сотрудниками, работающими в организации Б, в А может возникнуть потребность предоставить своим сотрудникам обратную *связь* для работы с электронной почтой и получения доступа к файлам. Однако, если эти сотрудники работают с компьютеров, входящих в состав внутренней сети организации Б, в которой используется *динамическая NAT* для скрытия адресов внутренних систем, это окажется невозможным. Если в вашей организации предпочтение отдается использованию *VPN* именно таким образом, следует проверить возможности программного обеспечения *VPN*.

Управление пользовательскими *VPN*

Управление пользовательскими *VPN*, главным образом, заключается в управлении пользователями и их компьютерами. При разделении сотрудников необходимо выполнять соответствующие процедуры по управлению пользователями.

Разумеется, на компьютерах пользователей должны устанавливаться правильные версии программного обеспечения *VPN* и реализовываться соответствующие конфигурации. Если компьютеры принадлежат организации, это *программное обеспечение* является стандартным компонентом для каждого компьютера. Если организация разрешает сотрудникам использовать *VPN* со своих домашних компьютеров, ей понадобится увеличить общий *уровень поддержки* этих пользователей, так как различные компьютеры и поставщики услуг интернета могут требовать наличие различных конфигураций.

Одним из ключевых аспектов пользовательской *VPN*, о котором не следует забывать, является установка хорошей антивирусной программы на компьютере пользователя. Этот программный пакет должен обеспечивать регулярное

обновление своих баз (по крайней мере, ежемесячно) для противостояния вирусам и "*троянским коням*", проникающим на *компьютер* пользователя.

Развертывание узловых сетей *VPN*

Узловые *виртуальные частные сети* используются организациями для подключения к удаленным узлам без применения дорогостоящих выделенных каналов или для соединения двух различных организаций, между которыми необходима *связь* для осуществления информационного обмена, связанного с деятельностью этих организаций. Как правило, *VPN* соединяет один *межсетевой экран* или пограничный *маршрутизатор* с другим аналогичным устройством (см. рис. 4).

Чтобы инициировать соединение, один из узлов осуществляет попытку передать трафик другому узлу. Вследствие этого на обоих противоположных узлах соединения *VPN* инициируется *VPN*. Оба конечных узла определяют параметры соединения в зависимости от политик, имеющихся на узлах. Оба сайта будут аутентифицировать друг друга посредством некоторого общего предопределенного секрета либо с помощью сертификата с открытым ключом. Некоторые организации используют узловые *VPN* в качестве резервных каналов связи для арендуемых каналов.

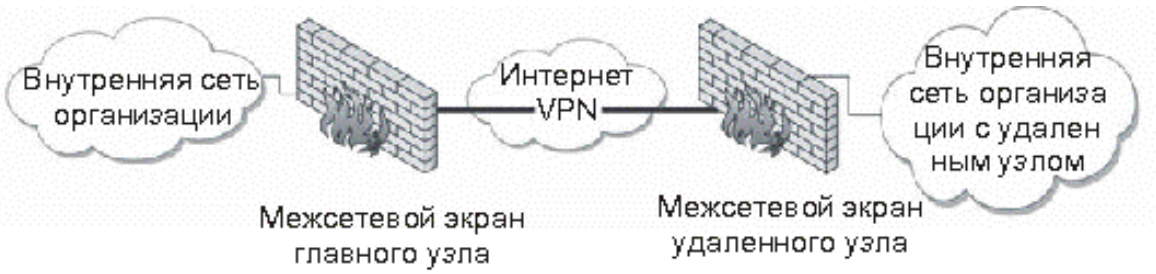


Рис. 4. Межузловое соединение *VPN*, проходящее через интернет

Преимущества узловых *VPN*

Как и в случае с пользовательскими *VPN*, основным преимуществом *узловой VPN* является экономичность. Организация с небольшими, удаленными друг от друга офисами может создать виртуальную частную *сеть*, соединяющую все удаленные офисы с центральным узлом (или даже друг с другом) со значительно меньшими затратами. *Сетевая инфраструктура* также может быть применена значительно быстрее, так как в удаленных офисах могут использоваться локальные *ISP* для каналов *ISDN* или *DSL*.

На базе политики организации могут быть разработаны правила, определяющие, каким образом удаленные сайты будут подключаться к центральному сайту или друг к другу. Если узловая *VPN* предназначена для соединения двух организаций, то на *доступ* ко внутренним сетям и компьютерным системам могут налагаться строгие ограничения.

Проблемы, связанные с узловыми *VPN*

Узловые *VPN* расширяют *периметр безопасности* организации, добавляя новые *удаленные узлы* или даже удаленные организации. Если уровень безопасности *удаленного узла* невелик, *VPN* может позволить злоумышленнику получить *доступ* к центральному узлу и другим частям внутренней сети организации. Следовательно, необходимо применять строгие политики и реализовывать функции аудита для обеспечения безопасности организации в целом. В случаях, когда две организации используют узловую *VPN* для соединения своих сетей, очень важную роль играют политики безопасности, установленные по обе стороны соединения. В данной ситуации обе организации должны определить, какие данные могут передаваться через *VPN*, а какие – нет, и соответствующим образом настроить политики на своих межсетевых экранах.

Аутентификация узловых *VPN* также является важным условием для обеспечения безопасности. При установке соединения могут использоваться произвольные секреты, но один и тот же общий секрет не должен использоваться для более чем одного соединения *VPN*. Если предполагается использовать сертификаты с открытыми ключами, необходимо создать процедуры для поддержки изменения и отслеживания срока действия сертификатов.

Как и в случае с пользовательскими *VPN*, *сервер VPN* должен поддерживать *дешифрование* и *шифрование VPN*-трафика. Если уровень трафика высок, *сервер VPN* может оказаться перегруженным. В особенности это относится к ситуации, когда *межсетевой экран* является *VPN*-сервером, и имеет *место интернет*-трафик большого объема.

Наконец, необходимо обдумать вопросы, связанные с адресацией. Если узловая *VPN* используется внутри одной организации, в ней необходимо наличие одинаковой схемы адресации для всех узлов. В данном случае *адресация* не представляет какой-либо сложности. Если же *VPN* используется для соединения двух различных организаций, необходимо предпринять меры для предупреждения любых конфликтов, связанных с адресацией. На рис. 5 отражена возникшая конфликтная ситуация. Здесь обе организации используют части одного и того же частного адресного пространства (*сеть* 10.1.1.x).

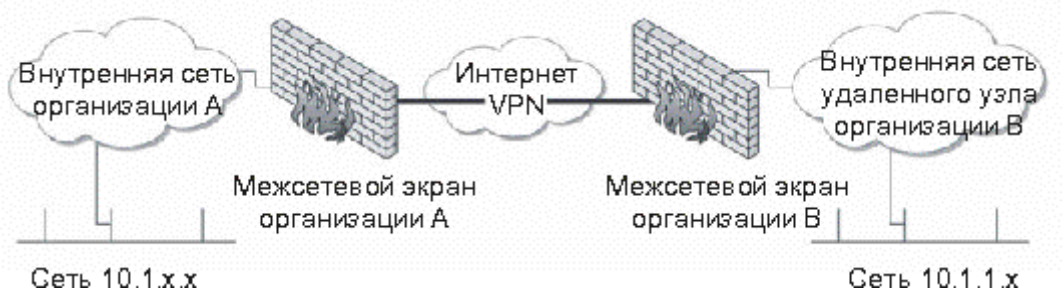


Рис. 5. Узловая VPN может вызывать конфликты, связанные с адресацией

Очевидно, что схемы адресации будут конфликтовать друг с другом, и *маршрутизация* трафика не будет функционировать. В данном случае каждая сторона соединения *VPN* должна выполнять трансляцию сетевых адресов и

переадресовывать системы другой организации на их собственную схему адресации (см. рис. 6).

Управление узловыми *VPN*

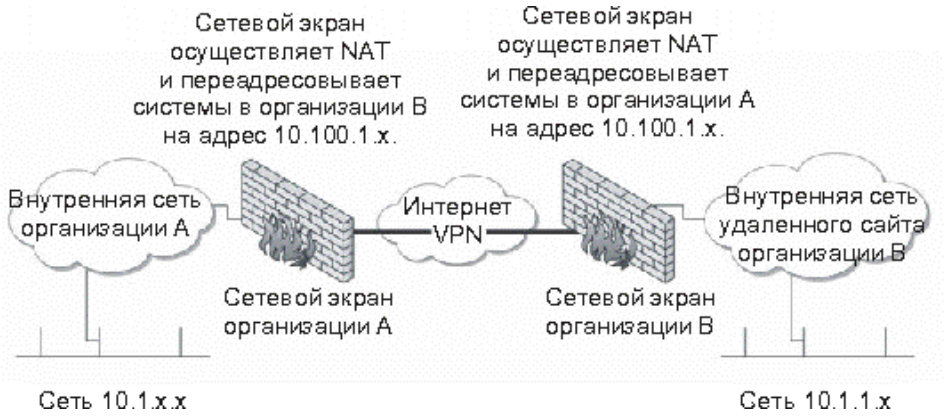


Рис. 6. Узловая VPN использует NAT для предотвращения конфликтов адресации

При осуществлении контроля над маршрутизацией могут понадобиться дополнительные функции по управлению. На маршрутизаторах внутренних сетей потребуется создать маршруты к удаленным сайтам. Эти маршруты, наряду с управлением схемой адресации, должны четко документироваться во избежание непреднамеренного удаления маршрутов в процессе управления маршрутизатором.

Понятие стандартных технологий функционирования *VPN*

Сеть VPN состоит из четырех ключевых компонентов:

- Сервер *VPN*.
- Алгоритмы шифрования.
- Система аутентификации.
- Протокол *VPN*.

Эти компоненты реализуют соответствие требованиям по безопасности, производительности и способности к взаимодействию. То, насколько правильно реализована *архитектура VPN*, зависит от правильности определения требований.

Определение требований должно включать в себя следующие аспекты:

- Количество времени, в течение которого необходимо обеспечивать защиту информации.
- Число одновременных соединений пользователей.

- Ожидаемые типы соединений пользователей (сотрудники, работающие из дома или находящиеся в поездке).
- Число соединений с удаленным сервером.
- Типы сетей VPN, которым понадобится соединение.
- Ожидаемый объем входящего и исходящего трафика на удаленных узлах.
- Политика безопасности, определяющая настройки безопасности.

Сервер *VPN*

Сервер VPN представляет собой *компьютер*, выступающий в роли конечного узла соединения *VPN*. Данный *сервер* должен обладать характеристиками, достаточными для поддержки ожидаемой нагрузки. Большая часть производителей программного обеспечения *VPN* должна предоставлять рекомендации по поводу производительности процессора и конфигурации памяти, в зависимости от числа единовременных *VPN*-соединений. Следует обеспечить наличие системы с соответствующими параметрами, а также позаботиться о ее дальнейшей модернизации. Может потребоваться создание нескольких серверов *VPN*, чтобы обеспечить поддержку ожидаемой нагрузки. В данном случае ожидаемые *VPN*-соединения должны как можно скорее распределяться между системами.

Некоторые производители включают в свои системы методы обхода ошибок и разрешают наличие избыточных серверов *VPN*. Обход ошибок может не подразумевать распределение нагрузки, поэтому соединения могут по-прежнему требовать распределения между серверами. Это обстоятельство необходимо принимать во внимание при построении систем.

VPN-сервер должен быть расположен в сети. *Сервер* может быть межсетевым экраном или пограничным маршрутизатором (см. рис. 7), что упрощает *размещение VPN*-сервера. В качестве альтернативы *сервер* может являться и отдельной системой. В этом случае *сервер* должен быть расположен в выделенной демилитаризованной зоне (*DMZ*) (см. рис. 8). В идеальном случае демилитаризованная зона *VPN* должна содержать только *VPN-сервер* и быть отдельной от *DMZ* интернета, содержащей веб-серверы и почтовые серверы организации. Причиной является то, что *VPN-сервер* разрешает *доступ* ко внутренним системам авторизованным пользователям и, следовательно, должен рассматриваться как *объект* с большей *степенью доверия*, нежели почтовые и веб-серверы, *доступ* к которым может быть осуществлен лицами, не пользующимися доверием. Демилитаризованная зона *VPN* защищается набором правил межсетевого экрана и разрешает передачу только того трафика, который требует *VPN*.

Рис. 7. Архитектура сети VPN, в которой межсетевой экран является VPN-сервером

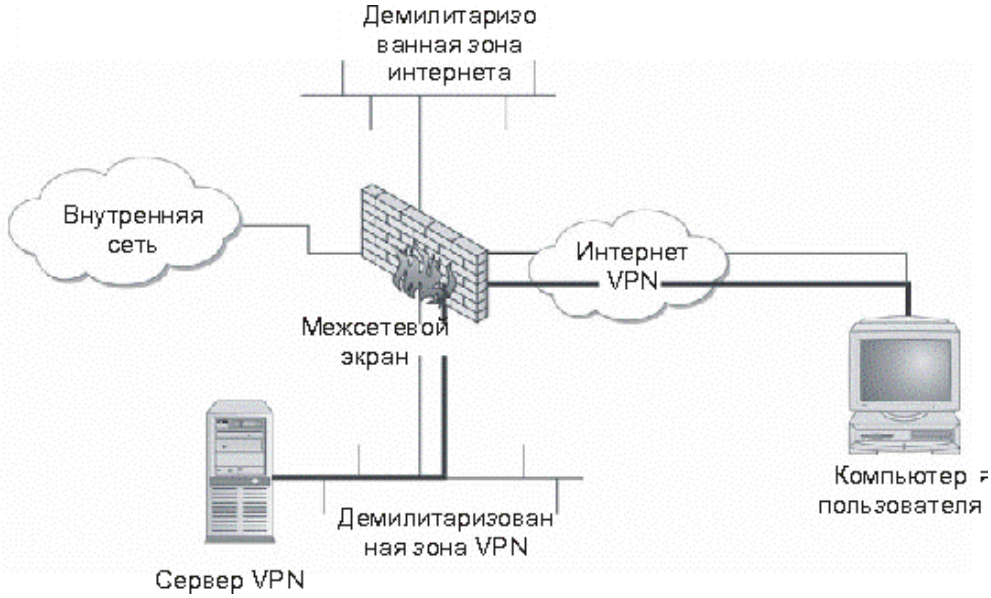


Рис. 8**.** Архитектура сети *VPN* для отдельного сервера *VPN*

Правила политики межсетевого экрана для демилитаризованной зоны *VPN* определены в табл. 1. Здесь содержатся правила, необходимые для демилитаризованной зоны интернета и демилитаризованной зоны *VPN*.

Правила 1, 2 и 3 относятся к демилитаризованной зоне *VPN*. Правило 1 позволяет клиентам *VPN* осуществлять *доступ* к серверу *VPN* с использованием любой службы, требуемой программным обеспечением *VPN*. Правило 2 разрешает *VPN*-серверу осуществлять маршрутизацию этих соединений во внутреннюю *сеть*. Правило 3 исключает соединение демилитаризованной зоны интернета с демилитаризованной зоной *VPN*, изолируя демилитаризованную зону *VPN* от систем в *DMZ* интернета, пользующихся меньшим доверием.

Таблица 1.

Правила политики межсетевого экрана, включающие демилитаризованную зону VPN

Номер правила	**Исходный IP**	**Конечный IP**	**Служба**	**Действие**
1	Любой	*VPN*-сервер	Служба *VPN*	Принятие.
2	*VPN*-сервер	Внутренняя сеть	Любой	Принятие
3	Любой	*VPN*-сервер	Любой	Отклонение

4	Любой	Веб-сервер	*HTTP*	Принятие
5	Любой	Почтовый сервер	*SMTP*	Принятие
6	Почтовый сервер	Любой	*SMTP*	Принятие
7	Внутренняя сеть	Любой	*HTTP, HTTPS, FTP, telnet, SSH*	Принятие
8	Внутренняя *DNS*	Любой	*DNS*	Принятие
9	Любой	Любой	Любой	Сброс

Алгоритмы шифрования

Алгоритм шифрования, используемый в *VPN*, должен быть стандартным мощным алгоритмом шифрования. Возникает вопрос: какая же система шифрования самая лучшая? Вообще, все стандартные и мощные алгоритмы могут эффективно использоваться при построении *VPN*. Различные производители отдают предпочтение различным алгоритмам, в зависимости от ограничений реализации продукта, аспектов, связанных с лицензированием, и предпочтений по программированию. Приобретая программный пакет *VPN*, следует выслушать комментарии специалистов и убедиться в том, что производитель использует мощный *алгоритм* шифрования.

Следует заметить, что выбор алгоритма не имеет принципиального значения, если он будет стандартным и в достаточной степени мощным. Гораздо больше влияет на общий уровень безопасности реализация системы. Неправильно реализованная система может сделать бесполезным самый мощный *алгоритм* шифрования. Приняв во внимание, сказанное выше, давайте изучим риски, связанные с использованием *VPN*.

Для того чтобы получить *доступ* к информации, передаваемой через *VPN*, *злоумышленник* должен:

- захватить весь сеанс соединения, т. е. разместить устройство прослушивания между противоположными концами соединения в том месте, через которое должен передаваться весь трафик *VPN*;
- использовать большие вычислительные мощности и большое количество времени для перехвата ключа с помощью грубой силы и для дешифрования трафика.

Злоумышленнику гораздо проще использовать имеющуюся *уязвимость* на компьютере пользователя либо украсть портативный *компьютер*, например, в аэропорту. Если *информация* не представляет собой особой важности, в *VPN* можно использовать любой широко распространенный, мощный *алгоритм* шифрования.

Система аутентификации

Третьим компонентом архитектуры *VPN* является *система аутентификации*. Как уже говорилось ранее, *система аутентификации VPN* должна быть двухфакторной. Пользователи могут проходить аутентификацию с использованием того, что они знают, того, что у них есть или с помощью данных о том, кем они являются. При использовании пользовательских *VPN* тдается предпочтение первым двум вариантам.

Хорошей комбинацией средств аутентификации являются смарт-карты в паре с персональным идентификационным номером или паролем. Производители программного обеспечения, как правило, предоставляют организациям на выбор несколько систем аутентификации. В данном перечне присутствуют ведущие производители смарт-карт.

Использование смарт-карт повлечет за собой увеличение стоимости использования *VPN* для каждого пользователя. Несмотря на то, что это обстоятельство повысит *стоимость* использования соединения, обеспечение более высокого уровня защиты этого стоит.

Если в организации предпочитают при использовании *VPN* полагаться только на пароли, они должны быть мощными (как *минимум*, сочетание из восьми букв, цифр и специальных символов) и регулярно изменяться (каждые 30 дней).

Протокол *VPN*

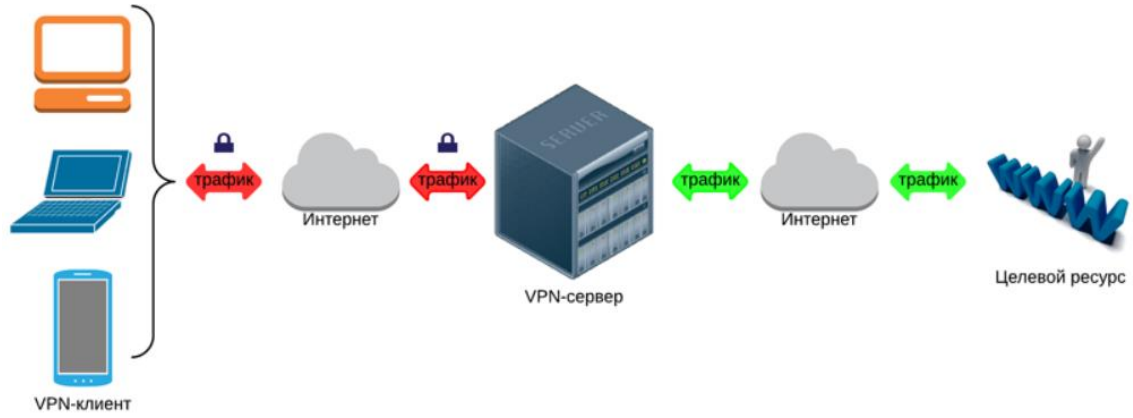
Протокол *VPN* определяет, каким образом система *VPN* взаимодействует с другими системами в интернете, а также уровень защищенности трафика. Если рассматриваемая организация использует *VPN* только для внутреннего информационного обмена, вопрос о взаимодействии можно оставить без внимания. Однако если организация использует *VPN* для соединения с другими организациями, собственные протоколы использовать, скорее всего, не удастся. В разговоре об алгоритме шифрования было упомянуто, что внешние окружающие факторы могут оказывать большее влияние на *безопасность системы*, чем *алгоритм* шифрования. Протокол *VPN* оказывает влияние на общий уровень *безопасности системы*. Причиной этому является тот факт, что протокол *VPN* используется для обмена ключами шифрования между двумя конечными узлами. Если этот обмен не защищен, *злоумышленник* может перехватить ключи и затем расшифровать трафик, сведя на нет все преимущества *VPN*.

При соединении рекомендуется использовать *стандартные протоколы*. В настоящее время *стандартным протоколом* для *VPN* является *IPSec*. Этот протокол представляет собой *дополнение* к *IP*, осуществляющее инкапсуляцию и *шифрование* заголовка *TCP* и полезной информации, содержащейся в пакете. *IPSec* также поддерживает обмен ключами, удаленную аутентификацию сайтов и согласование алгоритмов (как алгоритма шифрования, так и хэш-функции). *IPSec* использует *UDP-порт* 500 для начального согласования, после

чего используется *IP*-протокол 50 для всего трафика. Для правильного функционирования *VPN* эти протоколы должны быть разрешены.

Обзор существующих *VPN*-сервисов

В заключение еще раз вкратце опишем принцип действия *VPN* и рассмотрим некоторые имеющиеся *VPN* - сервисы. При обычном серфинге в Сети Ваш *IP*-адрес и местоположение доступны сторонним пользователям и сервисам, а сам канал передачи данных открыт для перехвата. При установке на свое устройство *VPN*-клиента весь трафик направляется через специальный сервер (туннель) в зашифрованном виде, что не даст возможность хакерам "вытащить" из него персональные данные (логины, пароли, данные банковских карт и т.д.). Кроме того, Вам присваивается некий виртуальный *IP*-адрес (можно стать даже пользователем другой страны), а реальный *IP* и местоположение скрываются, что не позволяет выследить Вас, а также обеспечивает доступ к заблокированным в Вашем регионе сайтам.



В настоящее время *VPN*-сервисов достаточно много. Выделим самые популярные, удобные и интересные.

CyberGhost VPN – позволяет использовать сервера в Европе и США, а данные шифруются мощным алгоритмом AES 256-бит. Пользоваться этим клиентом достаточно просто: выберите нужный (и доступный) сервер из списка и активируйте соединение, присутствует возможность ручного выбора IP-адреса. Для удобства предусмотрена карта мира с Вашим виртуальным местоположением. Бесплатная редакция этого *VPN* имеет некоторые ограничения по количеству используемых шлюзов, но для анонимности хватит и тех, что доступны. Реализованы версии для *Windows, Mac OS X, Android и iOS.*

VPN Browser Globus. Это целый браузер на основе *VPN*-технологии, защищающий от слежения и перехвата информации, позволяющий обойти ограничения доступа и утечки данных в общественных *WI-FI* сетях. Весь интернет-трафик пользователя надежно шифруется, *IP*-адрес изменен на адрес серверов разработчика. В остальном же обычный web-обозреватель со всеми необходимыми функциями и возможностями.

Avast SecureLine VPN. Фирменный *VPN*-клиент от популярного разработчика антивирусных продуктов. Как и другие подобные утилиты, этот инструмент анонимизирует все Ваши действия в Интернете, а также делает невидимыми для шпионов учетные данные, сообщения электронной почты, мгновенные сообщения и операции с кредитными картами как при подключениях через *Wi-Fi*, так и при подключениях других типов, обеспечивая тем самым полную безопасность использования интернета. Задействованы сервера в США, Великобритании, Германии, Нидерландах, Чехии, Южной Корее, Сингапуре и других странах. Имеются клиенты для *Windows, Mac OS X, Android и iOS*. К сожалению, *Avast SecureLine VPN* - условно-бесплатная программа, пробный период которой ограничен всего тремя днями.

ZenVPN. Весьма удобный и не требующий настройки *VPN*-клиент, использующий шлюзы в Нидерландах, США, Франции, Индии и Сингапуре. Приложение разработано по принципу "установили и забыли", при этом общая конфиденциальность и безопасность подключения к *Wi-Fi* точкам находятся на самом высоком уровне. Бесплатно в *ZenVPN* предлагается 250 Мб перенаправленного трафика в день, что вполне достаточно для простого серфинга.

Hideninja. Это приложение ориентировано на мобильную систему *Android*, создавая защищенные соединения в Интернете для предотвращения утечки важной персональной информации: пароли, данные банковских карт, регистрационные идентификаторы и т.д. Ну и конечно, *Hideninja VPN* выполняет и еще одну важную функцию – обеспечение доступа к ресурсам, которые заблокированы в Вашем регионе. Стать пользователем другой страны очень просто: выберите локацию (всего их 9) и подтвердите перенаправление трафика, кликнув по иконке щита на экране устройства. Бесплатная версия ограничена только 3 локациями, которые периодически меняются. Имеются клиенты *Hideninja VPN* для *iOS, Windows, Mac OS X*, а также расширения для браузеров *Chrome* и *Firefox*. Перед использованием сервиса нужно пройти простенькую регистрацию.

PureVPN. Достаточно мощный *VPN*-сервис, позволяющий получить доступ ко всем заблокированным в Вашем регионе ресурсам и обезопасить свои персональные данные от утечки при использовании общественных и незащищенных сетей. Особенностью *PureVPN* является достаточно обширная география расположения серверов: в арсенале сервиса более 450 серверов, обеспечивающих доступ к более чем 80 000 *IP*-адресов в более чем 100 странах. Минусом приложения является ограничение использования трафика: бесплатно можно перенаправить только 2 Гб данных после чего придется переходить на платную подписку. Так что этот клиент можно порекомендовать для временного применения. Реализованы версии для *Windows, Mac OS X, Android* и *iOS*

Private Tunnel VPN. Это еще один представитель *VPN*-технологии, имеющий в своем распоряжении клиенты для платформ *Windows, Mac OS X, Android* и *iOS*. Программа проста в настройке и использовании, однако выбор шлюзов невелик: всего 8 локаций – три в США и по одному в Канаде, Швейцарии, Великобритании, Нидерландах, Швеции. К тому же бесплатно можно перенаправить только 500 Мб

данных, что позволяет всего лишь протестировать приложение. Перед использованием клиента нужно пройти простую регистрацию.

Rocket VPN. Удобное приложение для мобильных устройств на базе ОС *Android*, которое позволяет обеспечить безопасность и анонимность просмотра веб-сайтов, имеющих какие-либо территориальные ограничения, и предоставляет доступ к заблокированным сервисам в сети Интернет. Для выбора доступны защищенные *VPN*-серверы в Чехии, Голландии, Великобритании, Швеции, США, Сингапуре и Японии. К сожалению, и в этом клиенте не обошлось без лимитирования – бесплатно можно использовать только 500 Мб в месяц.

Vpn One Click. Этот сервис полностью оправдывает свое название: Вам нужно всего лишь кликнуть по флажку страны, на которую Вы хотите поменять свое местоположение. Предусмотрено 28 локаций по всему миру: Великобритания, Франция, Германия, Испания, Россия, Швеция, США, Канада, Чили, Панама, Гонконг, Сингапур, Индия, Япония, Австралия, Египет и другие. Если сами не можете определиться со страной, то есть иконка случайного выбора местоположения. Вот, собственно говоря, и все: других настроек нет. *Vpn One Click* является условно-бесплатной программой, ограничение в которой после триального периода – невозможность самостоятельно выбрать нужный сервер (автоматическое подключение к шлюзу по усмотрению программы).

VyprVPN. Грамотно продуманный и простой в использовании *VPN*-клиент, позволяющий подключаться к множеству серверов по всему земному шару. Разработчики не поленились и реализовали в приложении более 50 разных стран по всему миру с более чем 700 серверами и 200 000 *IP*-адресов. Так что при желании Вы можете менять свое местоположение даже на специфические локации. Однако и тут не обошлось без ограничений: бесплатно предлагается только 500 Мб перенаправленных данных в месяц. Реализованы версии для *Windows, Mac OS X, Android* и *iOS*.

Есть и другие *VPN*-клиенты, но, как Вы уже поняли, принцип действия у них один и тот же. Так что при выборе подходящего для Вас ориентируйтесь на наличие лимита использованного трафика, простоту настройки и визуальное внешнее оформление. При этом не забывайте, что ни один *VPN*-сервис не является 100%-ной панацеей от вирусов и хакерских атак, так что в любом случае нужно быть осторожным в Сети, не посещать сомнительные ресурсы, не открывать подозрительные файлы и проявлять интернет-грамотность.

Прокси-сервер

Прокси – посредническое звено между компьютером, который использует абонент, и системой интернет-серверов. Если не вдаваться в терминологию, это удаленный компьютер-посредник для выхода пользователя в Интернет. Его основные задачи заключаются в трансляции всех запросов пользователя в Сеть и отправке обратно полученных ответов.

Как работает прокси-соединение

Каждому компьютеру, с которого осуществляется выход в Сеть, присвоен уникальный *IP*-адрес. Его задача – идентификация интернет-пользователя. *IP*-адрес несет информацию о стране и регионе, номере интернет-провайдера и персонального компьютера в его сети. Прокси-серверам тоже присвоены уникальные *IP*-адреса. После подключения к прокси и передачи запросов в Сеть проверка покажет, что они исходят с сервера-посредника, а сам абонент сможет сохранять свое инкогнито (в случае работы с бесплатными серверами, на платных информация о клиенте сохраняется).

Для подключения к прокси понадобится выполнить настройки в браузере, который будет использоваться для отправки пользовательских запросов. Все последующие сетевые подключения будут выполняться на *IP*-адрес прокси-сервера. Когда потребуется обращение к какому-либо веб-ресурсу, локальным компьютером будет открыто соединение с прокси и совершен запрос. После проверки корректности запроса откроется соединение с ресурсом. Затем полученный ответ будет передан на компьютер абонента.

Зачем и кому нужен прокси

На сегодняшний день прокси-серверы используются в основном для сокрытия истинного *IP*-адреса. Причин для этого может быть несколько. Наиболее популярные – желание посетить сайт, доступ к которому заблокирован для вашего *IP*, и необходимость анонимной отправки почты.

Есть еще несколько поводов, когда нужен прокси:

- защита сети или локального компьютера от некоторых видов сетевых атак и необходимость защиты конфиденциальной информации;
- ограничение доступа пользователей к некоторым типам веб-ресурсов. Практикуется в компаниях, чтобы предотвратить нерациональное расходование рабочего времени сотрудниками;
- желание подключить к Интернету несколько компьютеров при наличии одного *IP*-адреса. Настройки могут быть выполнены таким образом, что от внешних машин будет скрыта информация о локальных машинах, видеть они смогут только посредника;

- потребность в экономии потребляемого трафика – полученная из Сети информация будет передана пользователю в компактном виде;
- необходимость снижения нагрузки на интернет-канал и обеспечение клиентам оперативного доступа к информации. Для таких случаев выполняется кэширование файлов и их последующее хранение на прокси-сервере.

Прокси можно найти на некоторых сайтах, где они выкладываются бесплатно. Второй вариант – закачать с помощью специального ПО, которое позволяет использовать фильтр по странам и тестировать скорость и работу прокси-сервера. И еще один способ – купить «уполномоченный» сервер на специальных сайтах.

Преимущества

Наибольшую популярность они получили в корпоративном сегменте – именно через них осуществляется выход в онлайн-сети из локальных сетей юридических лиц. Этому поспособствовали следующие преимущества:

- прокси-серверы поддерживаются абсолютным большинством известных веб-браузеров;
- осуществляется полный контроль доступа, удобный учет трафика, его фильтрация (в случаях интеграции с антивирусами);
- возможность работы с минимальными правами на любой операционной системе;
- отсутствие выхода в Интернет по другим протоколам значительно повышает безопасность обмена информацией в корпоративной сети.

Несмотря на возрастающую популярность некоторых сетевых протоколов, прокси-серверы продолжают преобладать на предприятиях. И это несмотря на появление сравнительно недорогих аппаратных маршрутизаторов (роутеров) с функцией *NAT* (*Network Address Translation* – «трансляция сетевых адресов». Это процесс перевода внутренних адресов во внешние адреса. Если данная функция не будет настроена, то роутер заблокирует доступ к любым портам всем входящим соединениям из глобальной сети Интернет, при настроенных же параметрах – будет разрешать.). В основном это связано с тем, что вышеуказанные маршрутизаторы не в состоянии обеспечить достаточного контроля над выходом в Интернет и фильтрации контента.

Виды прокси-серверов

Прозрачный – схема связи, перенаправляющая маршрутизатором часть трафика (или весь) на прокси-сервер. Достоинством такого типа связи является возможность клиента пользоваться всеми преимуществами прокси-сервера без выполнения каких-либо настроек. Это же является и недостатком, т. к. лишает пользователя выбора.

Обратный – прокси-сервер, используемый для восполнения баланса сетевой нагрузки между несколькими веб-серверами. Кроме того, его задача – повышать их безопасность. В таком случае обратному прокси-серверу отводится роль межсетевого экрана на прикладном уровне. При его использовании запросы

пользователей ретранслируются из внешней сети на один или несколько серверов, расположенных во внутренней сети.

Кроме того, что прокси делятся на прозрачный и обратный типы, их можно классифицировать следующим образом:

- ***HTTP*** – наиболее популярный и универсальный тип прокси, который может использоваться для решения широкого спектра задач. Большая часть современного ПО поддерживает работу с ним;
- ***Socks*** – тип прокси, с которым совместима далеко не каждая программа. Требует установки дополнительного ПО в браузеры, т. к. они не поддерживают *Socks* по умолчанию. (Сокращение от «*SOCKet Secure*») – сетевой протокол, который позволяет пересылать пакеты от клиента к серверу через *прокси*-сервер прозрачно (незаметно для них) и таким образом использовать сервисы за межсетевыми экранами (фаерволами);
- ***CGI*** – взаимодействие с ними осуществляется только через браузер, другое ПО не осуществляет поддержку *CGI* (*Common Gateway Interface* «общий – шлюза») – стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа». По сути позволяет использовать консоль ввода и вывода для взаимодействия с клиентом. Иное название этого типа прокси – анонимайзер;
- ***FTP*** – тип прокси, часто используемый в корпоративных сетях в качестве одной из составляющих единой системы защиты оборудования от внешних угроз.

VPN (Virtual Private Network) или **Виртуальная Частная Сеть** – это намного более продвинутая технология, нежели прокси-сервера. Только *VPN* позволяет быть уверенным в своей анонимности в сети, так как трафик прокси-серверов может быть без проблем перехвачен и конечный запрашиваемый ресурс, при наличии определенного программного обеспечения, сможет соединиться с вашим компьютером напрямую, минуя прокси. Это, разумеется, приведет к полной "деанонимизации".

VPN же таких недостатков лишен. Помимо обеспечения полной анонимности, в частной виртуальной сети весь трафик шифруется и сжимается, так что перехватить его просто не представляется возможным. Как вы могли заметить, алгоритм работы у *VPN* схож с прокси, однако на самом деле разница довольно велика.

При использовании *VPN* на вашем компьютере устанавливается специальная программа, которая автоматически перенаправляет весь ваш трафик на адрес *VPN*-сервера, предварительно его зашифровав. Сервер же, в свою очередь, расшифровывает его и отправляет по назначению. Затем данные проходят обратный путь к вам аналогичным образом, а конечный ресурс не имеет никакой возможности определить ваш компьютер, так как он общается только с *VPN*, который может быть расположен, где угодно.

Еще одним большим плюсом данной технологии является обязательное использование *SSL*-сертификатов в *VPN*. Такой подход не позволит бдительным

сетевым администраторам узнать какие страницы вы посещаете. Вместо этого они смогут увидеть лишь бесполезные мусорные данные, расшифровать которые у них точно не получится.

Подводя итог можно сделать небольшой вывод: если вам нужно просто получить доступ на заблокированный ресурс – прокси будет достаточно, однако если вы хотите защитить себе от перехвата данных и значительно повысить свою анонимность – ваш выбор *VPN*.

Когда использовать прокси-сервер, а когда *VPN?*

И ***VPN*** и прокси-серверы были разработаны для того, чтобы помочь сохранить анонимность пользователей в интернете. А также дать возможность получить доступ к сайтам, которые заблокированы для определенных регионов.

Перед тем, как использовать прокси серверы, нужно знать их преимущества и недостатки. Для некоторых ситуаций больше подходит прокси, а для некоторых *VPN*.

Использование прокси сервера для локальных подключений ***дает следующие преимущества***:

- Вы скрываете свой *IP*-адрес при базовой проверке.
- Используя прокси-серверы, вы скрываете свое географическое положение. Сайты и сервисы, которые вы посещаете, видят только местоположение прокси-сервера.
- В зависимости от того, как настроены, прокси-серверы они могут повысить вашу безопасность путем блокирования сайтов, распространяющих вредоносное программное обеспечение. Они также могут проверять контент на наличие вредоносных элементов перед его отправкой на ваш компьютер.
- Прокси-серверы могут быть использованы для доступа к географически ограниченным сервисам.
- В интернете есть много открытых бесплатных прокси-серверов, и некоторые из них предоставляют разнообразные полезные услуги.

С другой стороны, есть и ***отрицательные моменты использования прокс***и сервера для локальных подключений, которые следует учитывать:

- Прокси-серверы не шифруют ваш интернет-трафик.
- Ни ваш *IP*-адрес, ни ваше реальное местоположение не скрыто от более продвинутых методов обнаружения. При использовании прокси-сервера ваш интернет-трафик проходит через него. Это означает, что вредоносный прокси-сервер может видеть и контролировать все, что вы делаете в интернете. Это наносит ущерб конфиденциальности и безопасности.
- Прокси-серверы, как правило, отслеживают и фиксируют действия своих пользователей. В определенных случаях это может иметь негативные последствия.
- Если вы обращаетесь к защищенному зашифрованному сайту или интернет-сервису, через неправильно настроенный прокси-сервер, то он может

передать на компьютер данные в незашифрованном виде. Незашифрованная информация может быть перехвачена другими лицами.

- Перед тем, как использовать прокси сервер в *Chrome*, учтите, что в интернете есть много открытых бесплатных прокси-серверов, и многие из них являются ненадежными. А некоторые из них вообще вредоносные.

Перед тем, как использовать прокси-сервер, учтите, что ***применение VPN дает много преимуществ***:

- Весь сетевой трафик между вами и *VPN*-сервером передается в зашифрованном виде, благодаря чему третьей стороне практически невозможно увидеть, какие сайты вы посещаете;
- Виртуальные частные сети скрывают ваш реальный *IP*-адрес. Сайты и интернет-сервисы будут видеть только *IP*-адрес *VPN*-сервера;
- Виртуальные частные сети скрывают ваше реальное географическое положение. Отображается только местоположение *VPN*-сервера, который вы используете;
- Использование *VPN*-сервера может помочь обойти географические ограничения;
- Ваш сетевой трафик не может быть просмотрен, так как все передается в зашифрованном виде;
- Есть множество провайдеров *VPN*, которые предлагают не только платные услуги, но и бесплатные *VPN*-серверы.

Возможно, вам лучше использовать прокси сервер для локальных подключений. Так как ***виртуальные частные сети имеют некоторые недостатки***:

- *VPN*-серверы должны шифровать весь трафик, который проходит через них, и это может сказаться на производительности и скорости;
- При подключении к *VPN*-серверу каждый бит данных между вами и сервером шифруется. Тем не менее, эти данные расшифровываются на сервере *VPN*, поэтому он знает, что вы делаете в интернете. Очень важно, чтобы провайдеры *VPN* не хранили логи (Логин – имя учетной записи пользователя в любой форме, т. е. имя Вашего аккаунта) деятельности пользователей. В противном случае провайдер *VPN* будет знать, что вы делаете. Эти данные могут быть использованы другими организациями, которые получают санкционированный или несанкционированный доступ к ним;
- Надежные *VPN*-сервисы, как правило, стоят дороже, чем хороший прокси-сервер. Шифрование всего трафика означает, что *VPN*-сервер должен иметь мощное аппаратное обеспечение.

Таким образом, виртуальные частные сети лучше, чем прокси-серверы практически во всех аспектах. Виртуальные частные сети лучше в плане анонимности и безопасности. Все, что вы делаете в интернете, используя *VPN*-сервис, шифруется, и никто не может контролировать или отслеживать ваши

действия. Единственным серьезным недостатком является то, что виртуальные частные сети стоят гораздо дороже, чем прокси-серверы.

Если все, что вам нужно, это скрыть свой *IP*-адрес или реальное местоположение от сайта или интернет-сервиса, который выполняет только базовую проверку, тогда используйте прокси сервер для локальных подключений. Если вам нужна анонимность, безопасность и конфиденциальность, в таком случае следует воспользоваться *VPN*. Тем не менее, проверьте, что выбранный вами сервис не хранит логи вашей деятельности.

Базовое руководство по установке и настройке прокси-серверов

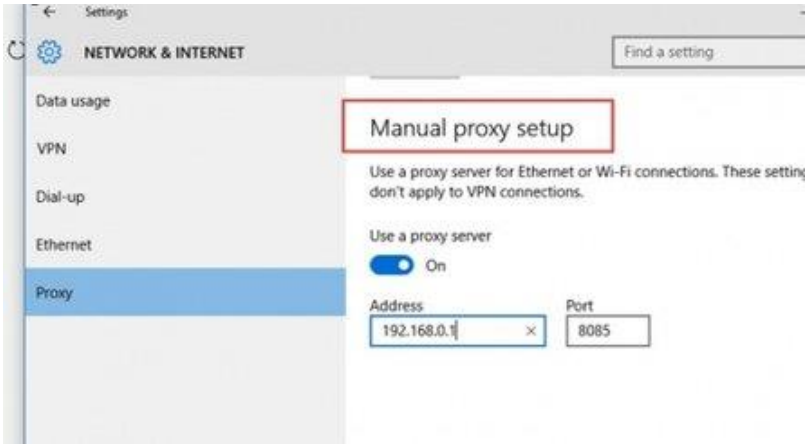
Перед тем, как использовать прокси сервер, нужно понимать, что интернет – это огромная гора данных. *Google* и другие поисковые системы пытаются собрать данные о пользователях, чтобы отфильтровать свою рекламу, продукты, сервисы и предложения для людей, которые возможно захотят купить или использовать их. Это по большей части здорово, поскольку люди, не возражают против того, чтобы получить продукты или сервисы, соответствующие их желаниям. Большая часть данных, передаваемых в виде *cooki*e-файлов, представляет собой ваше местоположение.

Но некоторые люди могут чувствовать, что предоставление реальных данных этим суперкомпьютерам и системам сбора делает их уязвимыми. Вот где использование прокси-сервера поможет скрыть эту информацию.

Конфигурирование прокси-сервера для работы с вашими настройками интернета на маршрутизаторе, а не просто прокси является отличным способом его использования. На сегодняшний день наиболее популярные браузеры, такие как *Internet Explorer, Firefox и Google Chrome*, допускают прокси-соединения.

Далее рассмотрим настройку браузера, чтобы использовать прокси для локальных серверов, начиная с *Microsoft Edge*, нового браузера для *Windows* 10, а также в *Internet Explorer*.

Microsoft Edge

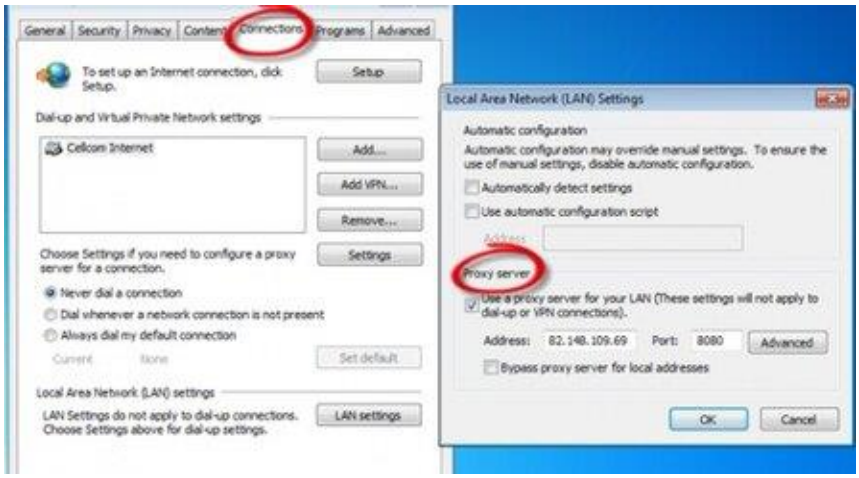


Откройте *Microsoft Edge*. Нажмите на кнопку «*Больше*» (*More*), которая расположена в правом верхнем углу окна. Выберите пункт «*Настройки*» (*Settings*),

который расположен в раскрывающемся меню в нижней части списка. Здесь выберите «*Дополнительные параметры*» (*Advanced Settings*).

Во всплывающем меню воспользуйтесь опцией «*Автоматическая и ручная настройка прокси*» (*Automatic and Manual Proxy Setup*). Введите информацию об используемом прокси-сервере в соответствующие поля.

Internet Explorer



Откройте *Internet Explorer*. Зайдите в раздел «*Настройки*», расположенный в правом верхнем углу страницы. Пролистайте до «*Свойства обозревателя*» (*Internet Options*). Нажмите на вкладку «*Подключения*» (*Connections*). Добавьте параметры используемого прокси-сервера в соответствующие поля.

Mozilla Firefox

Откройте браузер *Firefox*. В верхнем правом углу нажмите на кнопку меню. Затем на «*Дополнительные*» (*Advanced*). Выберите «*Сеть*» *(Network*). Нажмите на «*Настройки*» (*Settings*) во вкладке «*Соединение*» (*Connections*), «*Ручная настройка сервиса прокси*». Используйте параметры прокси для настроек. Добавьте прокси.

Google Chrome

Перед тем, как использовать прокси сервер в *Chrome*, откройте программу. Нажмите на кнопку меню в правом верхнем углу, выберите «*Настройки*» (*Settings*). Пролистайте до пункта «*Показать дополнительные настройки*» (*Show Advanced Settings*). Выберите пункт «*Сеть*» (*Network*), «*Изменить настройки прокси-сервера*» (*Change Proxy Settings*).
Нажмите «*Настройки*» (*Settings*), «*Настройка параметров локальной сети*» (*LAN settings*). Выберите пункт «*Использовать прокси-сервер для локальных подключений*». Введите параметры прокси-сервера.

6. Основы администрирования *Linux*

Поскольку мы используем персональный компьютер, надо уделить некоторое внимание задачам администрирования системы. Ведь у вас не будет системного администратора, к которому можно обратиться, столкнувшись с какой-нибудь проблемой. Хочется только с самого начала напомнить, что, в большинстве случаев, для конфигурирования системы необходимо иметь права суперпользователя *root*.

И еще одно предварительное замечание, которое поможет вам легче понять и освоить принципы администрирования *Linux*: любые настройки этой ОС могут быть выполнены путем редактирования файлов сценариев (или скриптов) и конфигурационных файлов, которые читаются скриптами. Причем и те, и другие (т. е. и скрипты, и конфигурационные файлы) являются простыми текстовыми файлами. Конечно, в *Linux* существуют различные специальные утилиты конфигурирования и администрирования системы (типа *linuxconf* или *printtoo*l), однако результаты работы этих программ все равно записываются в тех же конфигурационных файлах. Образно выражаясь, про *Linux* (и *UNIX* вообще) можно сказать "это почти целиком обработчик текста". Если с самого начала помнить об этой особенности, можно значительно легче освоить вопросы системного администрирования *Linux*.

Кстати, если вы хотите облегчить себе работу по редактированию конфигурационных файлов, сразу после инсталляции ОС *Linux* установите программу *Midnight Commander*. Это существенно облегчит вам поиск и редактирование конфигурационных файлов, так как можно будет пользоваться встроенным редактором этой программы (не говоря уж о том, что поиск нужного файла тоже сильно облегчается).

Основные задачи системного администрирования

К обязанностям системного администратора обычно относят следующие задачи:

- подключение и настройка аппаратных устройств;
- установка и обновление программного обеспечения;
- запуск и настройка общесистемных сервисов (конфигурирование системы);
- управление пользователями;
- управление процессами;
- распределение ресурсов;
- обеспечение безопасности.

Начнем с рассмотрения того, как происходит процесс загрузки ОС. Дело в том, что этот этап во многом определяет режим последующей работы системы и ее конфигурацию. Если вы умеете влиять на процесс загрузки, значит, вы уже сможете добиться желаемой конфигурации системы после загрузки.

Но для понимания процедуры начальной загрузки необходимо иметь самое общее представление о том, что такое процесс в системе, поскольку это понятие будет постоянно использоваться в дальнейшем.

В самом первом приближении можно считать, что процесс – это загруженная в оперативную память программа. Но это не совсем точно, правильнее было бы сказать, что "процесс выполняет программу". Дело в том, что в *Linux* вначале запускается процесс, который загружает в оперативную память программу из указанного ему файла и начинает ее выполнять. Это означает, что каждый процесс должен быть запущен (как говорят — "порожден") каким-то другим процессом. То есть для каждого процесса однозначно определен его "родитель" (или "предок"), для которого данный процесс является "дочерним" (или "потомком"). Если вы хотите увидеть "дерево" запущенных в вашей системе процессов, выполните команду *pstree*. Вывод этой команды позволяет увидеть, что "отцом" всех процессов в системе (или "корнем дерева процессов") является процесс *init*, который первым запускается после загрузки ядра.

Каждый процесс в системе имеет уникальный идентификатор – *PID*, назначаемый процессу при запуске. Процесс с идентификатором 1 выполняет программу *init*. Именно по этим идентификаторам система различает процессы. Каждый запущенный процесс в любой момент времени находится в одном из следующих состояний: активен (*R*), приостановлен (*T*) или "спит" (*S*). Текущее состояние процесса называют статусом процесса.

Кроме идентификатора и статуса для каждого процесса в специальных структурах ядра сохраняются следующие данные (приводимый ниже перечень является далеко не полным):

- полная командная строка запуска выполняемой процессом задачи;
- информация об отведенном процессу адресном пространстве;
- ссылка на текущий рабочий каталог и корневой каталог процесса (последний служит для ограничения доступа процесса к файловой структуре);
- таблица открытых процессом файлов;
- так называемое окружение процесса, т. е. перечень заданных для данного процесса переменных с их текущими значениями;
- атрибуты, определяющие права и привилегии процесса,
- таблица обработчиков сигналов;
- указание на родительский процесс;
- пользовательская маска (*umask*) или маска доступа – указание на то, какие права надо удалить при создании нового файла или каталога из стандартного набора прав, присваиваемых файлу (каталогу).

Поскольку *Linux* – система многозадачная, одновременно может быть запущено много процессов. Впрочем, слово "одновременно" здесь применено не совсем корректно, поскольку на самом деле в каждый момент времени выполняется только один процесс. (*Для точности следует заметить, что в многопроцессорных*

системах, на которых Linux тоже может работать, одновременно могут выполняться несколько процессов, но мы рассматриваем только однопроцессорные системы). Планировщик процессов выделяет каждому процессу небольшой квант времени и по истечении этого кванта передает управление следующему процессу. Кванты времени, выделяемые каждому процессу, так малы, что у пользователя создается иллюзия одновременного выполнения многих процессов. А для того, чтобы некоторые, наиболее важные процессы, получали больше процессорного времени, для каждого процесса установлен приоритет.

Пользователи могут "общаться" с процессами путем посылки им сигналов. Процессы тоже общаются друг с другом посредством сигналов. Когда мы нажимаем комбинацию клавиш <*Ctrl*>+<*C*>, чтобы завершить выполнение какой-то программы, мы фактически посылаем соответствующему процессу сигнал "Завершить работу". Завершаясь, процесс посылает родительскому процессу сигнал о своем завершении. Но бывают случаи, когда родительский процесс завершается раньше дочернего. Процессы, не имеющие родителя, называются "сиротами". "Сироты" автоматически усыновляются процессом *init*, который и принимает сигналы об их завершении. Если процесс-родитель по каким-то причинам не может принять сигнал о завершении дочернего процесса, то процесс-потомок превращается в "зомби" и получает статус *Z*. Процессы-зомби не занимают процессорного времени (т. е. их выполнение прекращается), но соответствующие им структуры ядра не освобождаются. Уничтожение таких процессов — одна из обязанностей системного администратора. Наконец, процесс может надолго "впасть в сон", прервать который не удается. Статус таких процессов обозначается символом *D*. Уничтожить их удается только при перезагрузке системы.

Особым видом процессов являются демоны. Вообще-то в них нет ничего особого. Это просто процессы, выполняющиеся в фоновом режиме, без вывода каких-либо данных на терминал. Демоны обычно используются для выполнения сервисных функций, обслуживания запросов от других процессов, причем не обязательно выполняющихся на данном компьютере.

Надо еще упомянуть, что процессы могут запускать ("внутри себя") отдельные нити (*thread*), или потоки. Нити – это параллельно выполняемые части одной программы, которые в *Linux* реализованы как процессы, запускаемые со специальным флагом. С точки зрения системы они отличаются от других процессов только тем, что для них не создается отдельное окружение, они выполняются в среде родительского процесса.

Как и любая другая операционная система, *Linux* нуждается в обслуживании, настройке и решении проблем.

Попытаемся охватить основные задачи, которые возникают перед администратором будь то сервера или домашнего компьютера. Вот основные задачи, которые мы рассмотрим:

- Удаленный доступ.
- Диагностика сети.
- Мониторинг ресурсов системы.
- Проверка работоспособности сервисов.
- Просмотр логов.
- Установка программного обеспечения.

С первоначальной настройкой сервера обычно проблем не возникает. Можно использовать одну из известных панелей управления, например, панель управления *VestaSP*, которая позволяет все установить и настроить автоматически, но вам нужно следить за показателями системы и перенести на сервер файлы.

Удаленный доступ к серверу *Linux*

Чаще всего веб-мастера и администраторы используют для удаленного доступа и загрузки файлов на сервер протоколы *SSH* и *FTP*.

SSH (Secure Shell) – это протокол удаленного управления компьютером с операционной системой *Linux*. В основном *ssh* используется для удаленного управления серверами через терминал.

FTP (File Transfer Protocol) – протокол передачи файлов по сети

По *SSH* вы не только передавать файлы, но и выполнять на сервере различные команды *Linux*. Протокол *FTP* позволяет лишь загружать файлы на сервер, перемещать и переименовывать их. Если кратко, то, например, чтобы перенести файлы сайта с одного сервера на другой, сначала мы создаем архив с помощью *tar*:

$ tar cvzf backup.tar.gz /папка/с/файлами

Когда архив готов, используем копирование *scp* для передачи его на сервер:
$ scp backup.tar.gz user@ip_*сервера:/var/www/public_html/*

Затем авторизуемся на сервере и распаковываем архив:
$ *ssh user@ip_*сервера
$ *cd /var/www/public_html/*
$ tar xvzf backup.tar.gz

После этого останется сменить владельца для распакованных данных на имя пользователя веб-сервера:

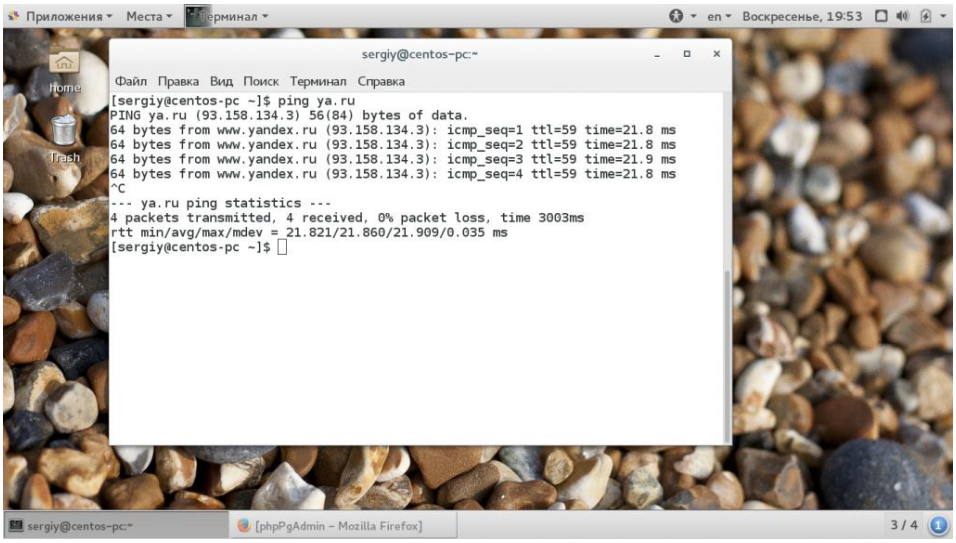
$ chown -R www-data /var/ww/public_html/project/

Для выполнения большинства действий по администрированию сервера вам надо знать как пользоваться утилитой *ssh*.

Диагностика сети Linux

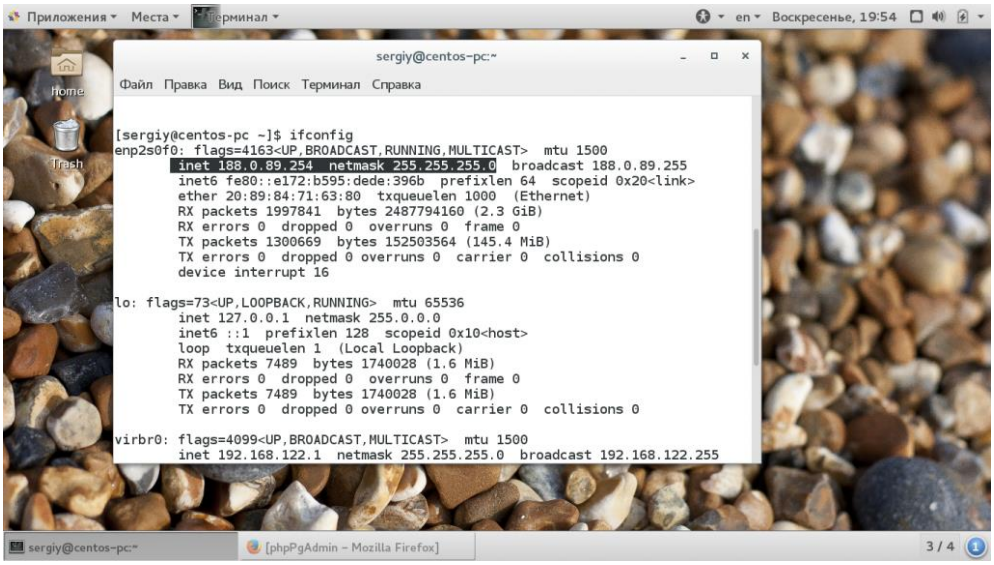
Это администрирование *Linux* серверов подходит больше для компьютеров, к которым у вас есть физический доступ, но может в некоторых случаях полезным и на сервере. Самый простой способ проверить есть ли доступ к сети на компьютере, это выполнить команду *ping*:

ping ya.ru



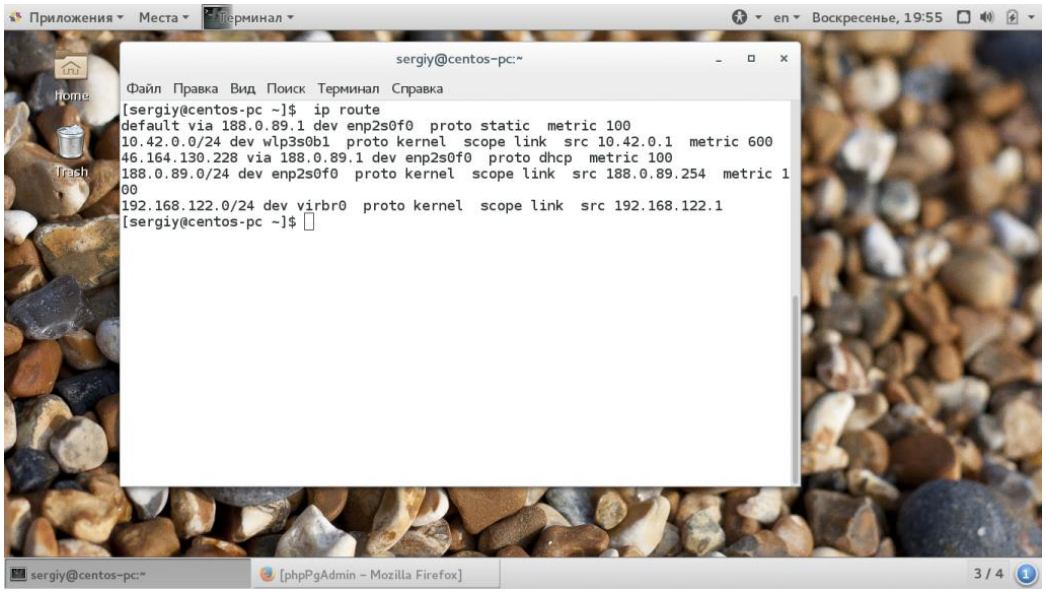
Если команда правильно отрабатывает, и вы видите передачу пакетов на удаленный узел, значит все хорошо. Если же нет, хотелось бы понять причину. Проверьте, указан ли *ip*- адрес и маска сети для этого подключения:

Ifconfig



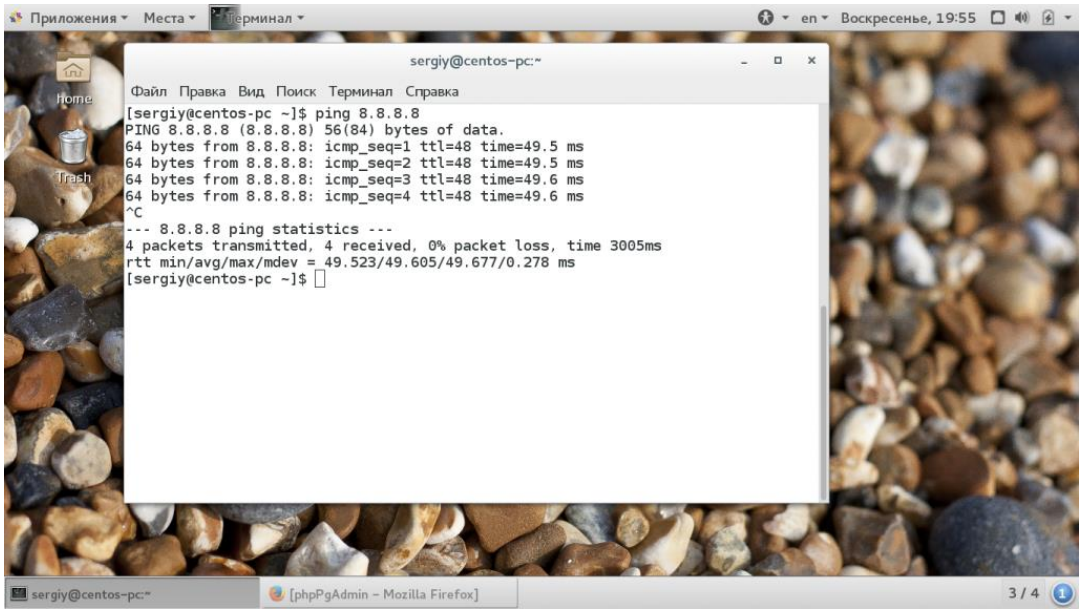
Убедитесь, что правильно задан шлюз доступа к сети:

ip route



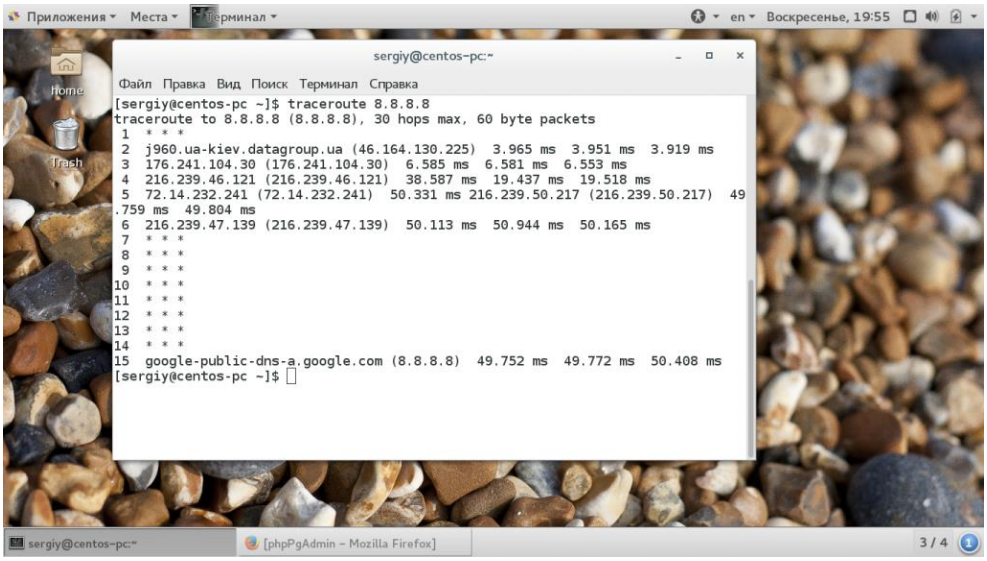
Обычно, это может подсказать вам, что конфигурация сети выполнена неверно, например, компьютер не получает нужные данные по протоколу *DHCP* или заданы неправильные статические настройки. Также проблема может быть в *DNS*. Возможно, сеть есть, но сервер не может получить *ip*-адрес на основе доменного имени, для проверки вы можете выполнить *ping* какого-либо внешнего *ip*:

ping 8.8.8.8



Если же сеть не работает, и она настроена правильно, то можно еще попытаться узнать на каком узле обрывается соединение. Для этого используется команда *traceroute*:

traceroute 8.8.8.8

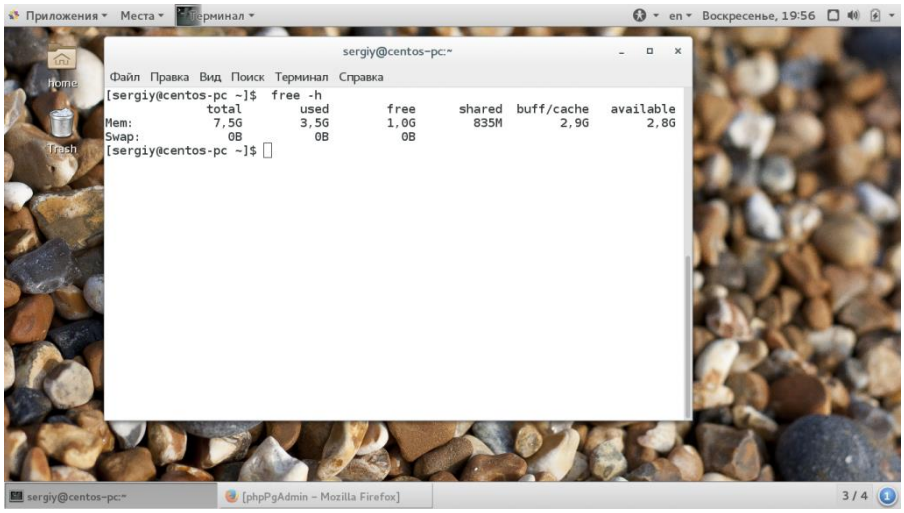


Все эти данные помогут понять в чем была ошибка и как ее решить.

Мониторинг ресурсов системы

Часто может случится, что сервер начинает работать очень медленно, веб-службы начинают очень долго отвечать на запросы и даже соединение по *SSH* работает медленно. Скорее всего, причиной этому может стать перегрузка ресурсов процессора или памяти. Если вся память будет занята, система будет сбрасывать данные на диск, в раздел подкачки, что тоже сильно замедляет работу сервера. Чтобы посмотреть сколько памяти осталось доступно используйте команду *free*:

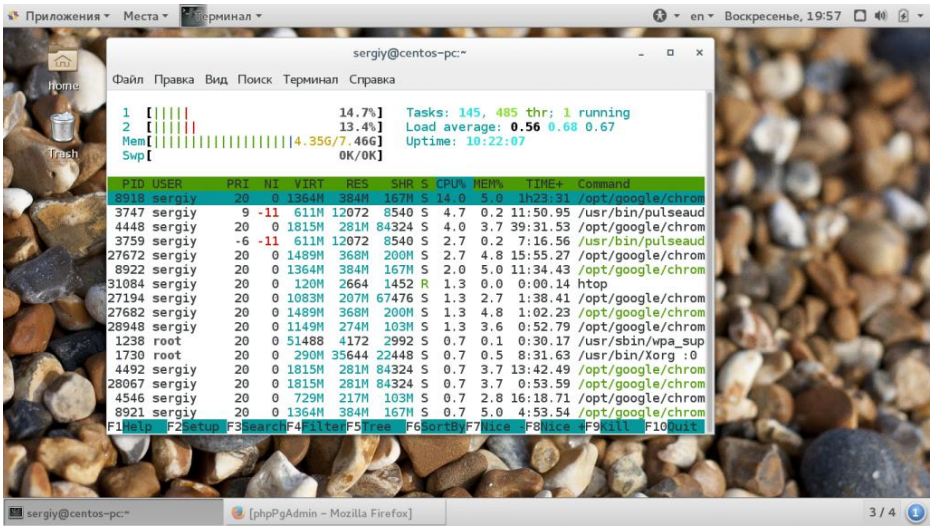
free –h

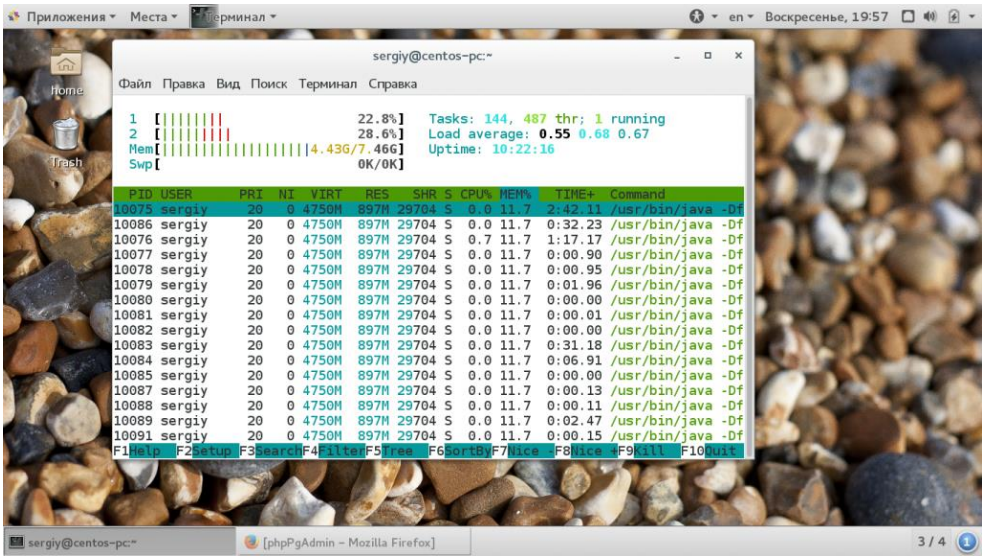


Естественно, что если свободно только 40-50 Мб, то этого системе очень мало и все будет работать очень медленно. Следующим шагом будет выяснить

какой процесс потребляет больше всего памяти, для этого можно использовать команду *htop*:

htop

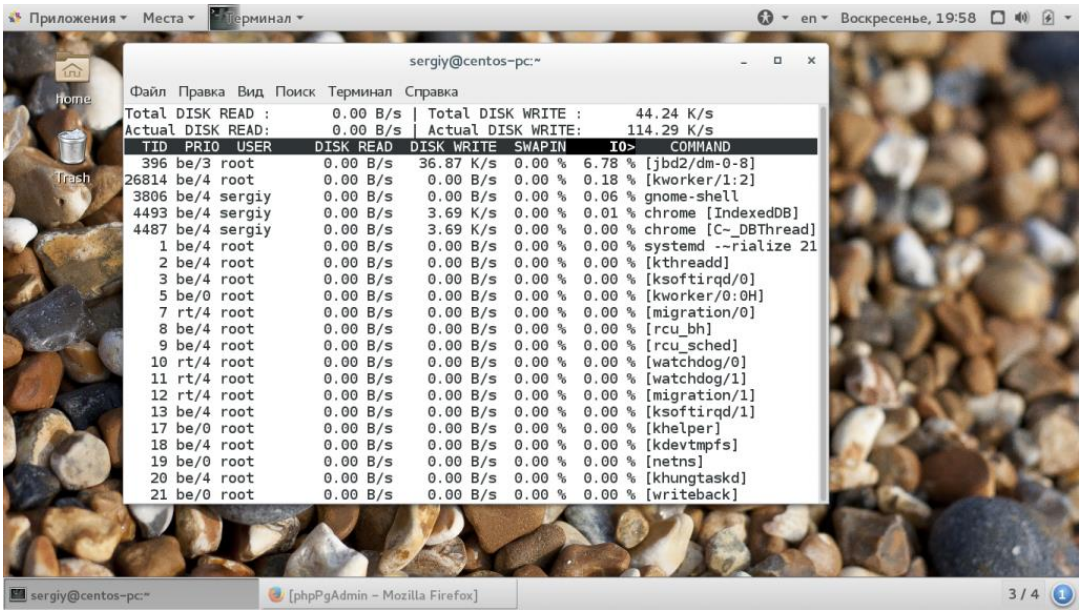




В утилите вы можете сортировать процессы по загрузке процессора, колонка %*CPU*% или по потреблению памяти %*MEM*%. Так вы можете очень просто понять в чем проблема и кто перегружает систему. Например, веб-сервер *Apache* потребляет слишком много памяти, поэтому, возможно, будет эффективнее использовать *Nginx*.

Также в некоторых случаях нас может интересовать загрузка диска *Linux* и какие именно процессы перегружают жесткий диск. Для этого применяется утилита *iotop*. Просто выполните утилиту без параметров:

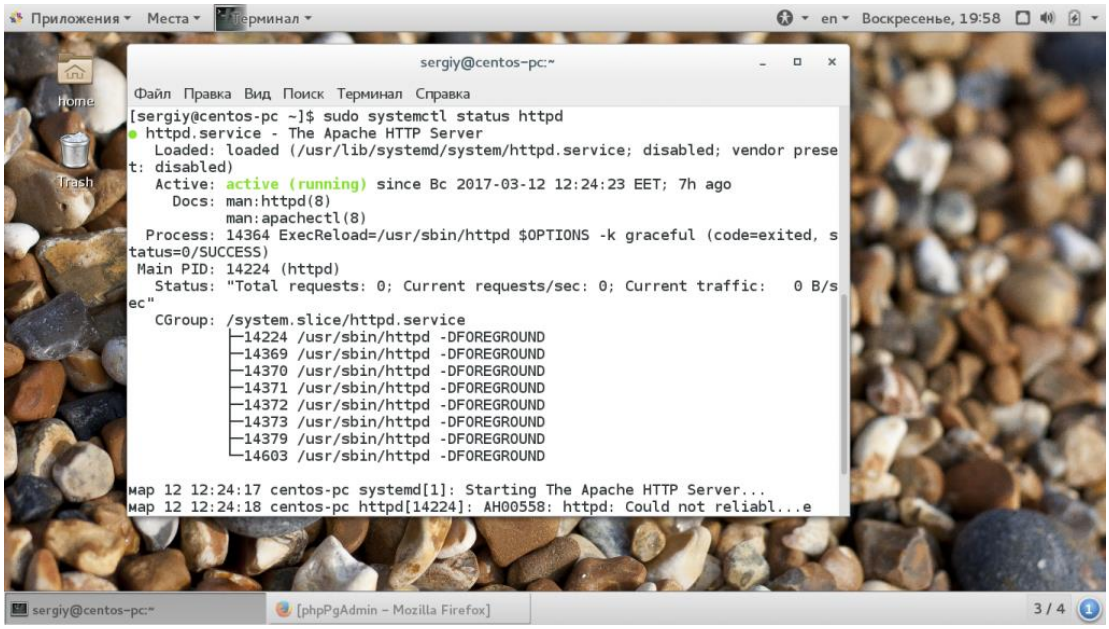
iotop



Проверка работоспособности серверов

В системное администрирование *Linux* также входит управление сервисами. Сейчас в большинстве дистрибутивов, в качестве системы инициализации используется *systemd*. Соответственно, управление службами *Linux* выполняется с помощью нее. Чтобы посмотреть запущена ли служба, например, веб-сервер *nginx*, выполните:

sudo systemctl status httpd



В многочисленном выводе утилиты вы должны увидеть сообщение

Active (running)

Это означает, что все хорошо и служба работает так как нужно.
Возможно, также вам придется перезапустить службу:

sudo systemctl restart httpd

Или запустить ее, если она не была запущена до этого:

sudo systemctl start httpd

Если служба не запустилась, то вы можете посмотреть информацию об этом с помощью команды *status* или же выполнить: *journalctl –xe*

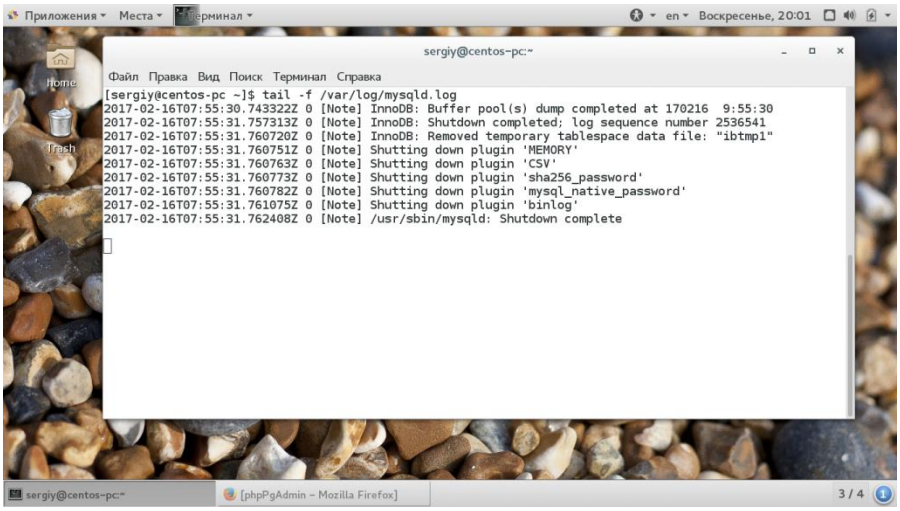
Просмотр логов

Лог-файл (файл журнала) сохраняет хронологию событий для операционной системы *Linux*, её приложений и служб. Файлы хранятся в виде обычного текста, чтобы их можно было легко читать.

Если какой-либо сервис или системный компонент не работает, то первое что нужно сделать – это смотреть логи. Если не помогает – включить режим отладки и смотреть логи. В 90% вы найдете ответ почему ничего не работает в логах программы. Логи всех служб и системные логи находятся в папке */var/log/*. Некоторые службы создают отдельные папки для своих файлов, например, */var/log/nginx* или */var/log/apache*.

Если в обычном логе вы не нашли решение, то можно переключить программу в режим отладки или включить отображение максимально подробной информации. Обычно это дается в конфигурационном файле программы. Приводить конкретные примеры нет смысла, поскольку у каждого сервиса все по-другому. Но рассмотрим несколько команд, которые вы можете использовать:

***tail -f /*путь/к/лог/файлу**



С помощью этой команды вы можете в реальном времени просматривать изменения в конце лог файла. Если опцию *-f* не указывать, то команда *tail* покажет десять последних строк из лога:

tail /путь/к/лог/файлу

Также для просмотра лог файла вы можете использовать любой текстовый редактор или утилиту *cat*.

Установка программного обеспечения

Установка программного обеспечения, одна из распространенных задач администрирования. В *Linux* большинство программ можно установить из официальных или сторонних репозиториев. Некоторые программы нужно собирать из исходников. Для установки софта из репозиториев используется пакетный менеджер. Существуют два основных пакетных менеджера, которые применяются на серверах, это *yum*, который используется в *CentOS* и *apt*, который применяется в ОС *Ubuntu*. Работают пакетные менеджеры похожим образом. Например, чтобы установить пакет в *Ubuntu* используйте такую команду:

sudo apt install имя_пакета

А в *CentOS/RedHat:*

sudo yum install имя_пакета

Для удаления программы используется команда *remove* вместо *install*. Но что еще более важно для серверов обновление программ. Никогда не отключайте автоматическое обновление, и старайтесь следить, чтобы система была в самом актуальном состоянии. Нужно обновлять все программные продукты, поскольку в них постоянно обнаруживаются новые уязвимости и следует получить вовремя исправления для них.

7. Системное администрирование с помощью сетевых инструментов *Linux*

В повседневные задачи системных администраторов входит работа с сетями и с подключённым к ним оборудованием. Нередко роль рабочего места администратора играет компьютер, на котором установлен какой-нибудь дистрибутив *Linux*. Утилиты и команды *Linux*, о которых пойдёт речь, включают в себя список инструментов различной сложности, которые предназначены для решения широкого спектра задач по управлению сетями и по диагностике сетевых неполадок.

В некоторых из рассматриваемых здесь примеров используются сокращения <*fqdn*> (*fully qualified domain name* – полное доменное имя). Встретив его, замените его, в зависимости от обстоятельств, на адрес интересующего вас сайта или сервера, например, на *server-name.company.com.*

Утилиты администратора

В некоторых из рассматриваемых здесь примеров вы столкнётесь с сокращением <*fqdn*> *(fully qualified domain name*, полное доменное имя). Встретив его, замените его, в зависимости от обстоятельств, на адрес интересующего вас сайта или сервера, например, на нечто вроде *server-name.company.com.*

Ping

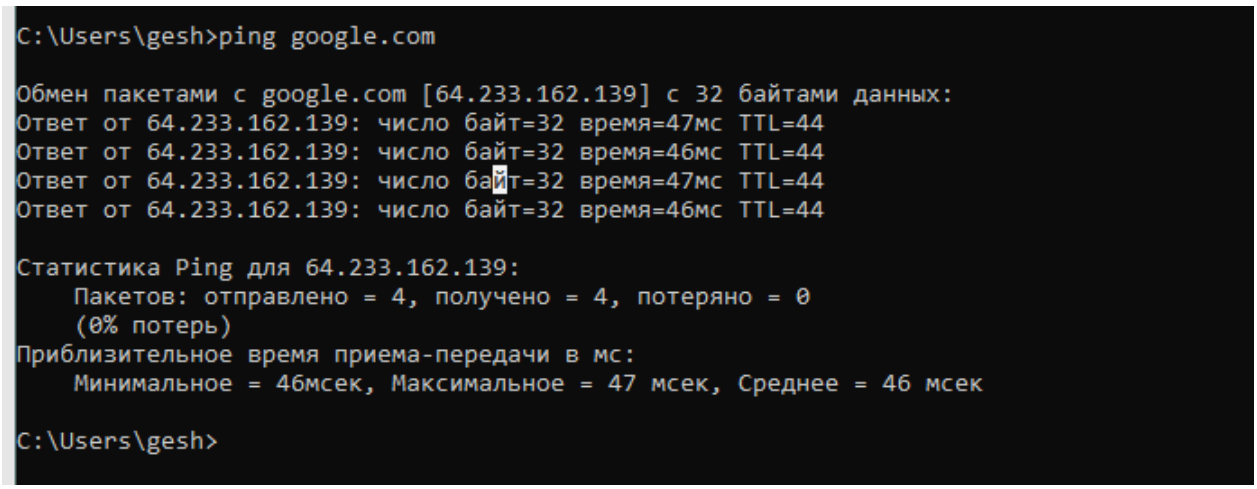
Утилита *ping*, как можно судить по её названию, используется для проверки связи между узлами сети, между компьютером, на котором её запускают, и другой системой. Эта утилита использует протокол *ICMP*, отправляя эхо-запросы, на которые отвечает удалённая система, получающая их. Использование *ping*, кроме того – это хороший способ проверки связности сети, проводимой в качестве первого шага диагностики сети при наличии неполадок. Команду *ping* можно использовать с адресами IPv4 и *IPv*6.

Примеры

IPv4: ping <ip address>/<fqdn>

IPv6: ping6 <ip address>/<fqdn>

Ping, кроме того, можно использовать для выяснения *IP*-адресов сайтов на основе их имён.



IP-адреса сайта *google.com*

Traceroute

Traceroute – это приятная утилита, которая позволяет исследовать маршруты передачи данных между компьютерами. В то время как команда *ping* направлена на то, чтобы выяснить, можно ли установить связь между двумя узлами сети, *traceroute* даёт сведения об *IP*-адресах маршрутизаторов, через которые проходят данные от вашей системы до конечной, например – до веб-сайта или сервера. Команда *traceroute* обычно применяется на втором шаге диагностики сети, после команды *ping*.

Пример

traceroute <ip address>/<fqdn>

Telnet

Утилита *telnet* позволяет связаться с удалённым компьютером по протоколу *Telnet* и взаимодействовать с ним, используя соответствующие команды.

Пример

Для организации сеанса Telnet-связи с другим компьютером используется следующая команда:

telnet <ip address>/<fqdn>

Netstat

Эта команда позволяет собирать сведения о сети и используется в ходе поиска и исправления сетевых неполадок, применяется для проверки данных о работе интерфейсов и портов, для исследования таблиц маршрутизации, для

изучения информации о работе протоколов. Эта команда непременно должна присутствовать в арсенале системного администратора.

Примеры

Для того чтобы получить список всех портов, находящихся в режиме прослушивания, воспользуйтесь такой командой:

```
netstat -l
```

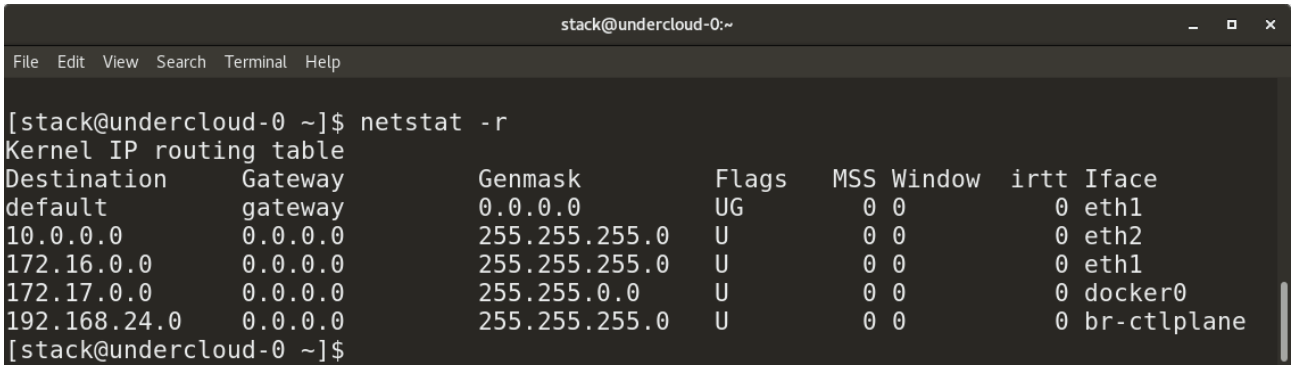
Следующая команда выводит сведения обо всех портах. Для того чтобы ограничиться только *TCP*-портами, нужно воспользоваться ключом *-at*, для того, чтобы получить данные об *UDP*-портах, используйте ключ *-au*.

```
netstat -atr
```

Для просмотра таблиц маршрутизации воспользуйтесь такой командой

```
netstat -r
```

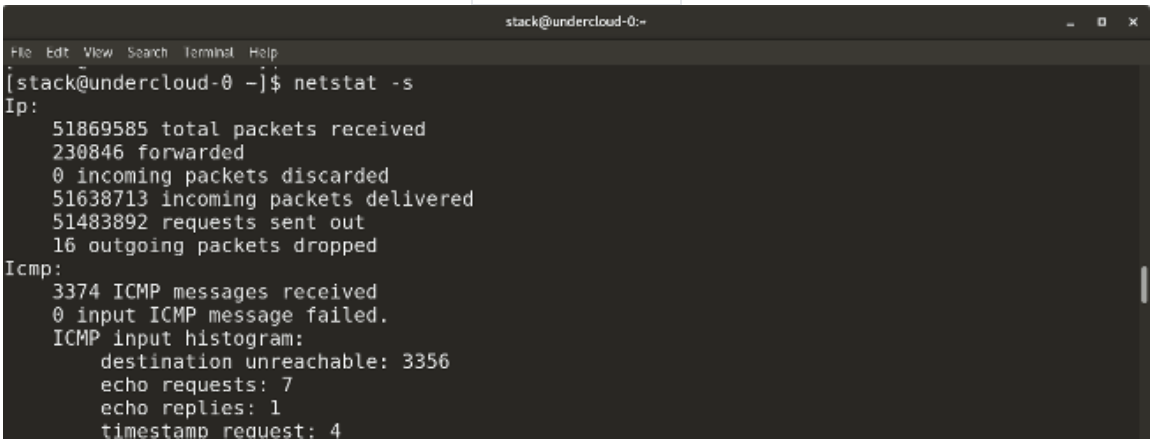
Вот как выглядит результат выполнения этой команды.



Сведения о таблице маршрутизации

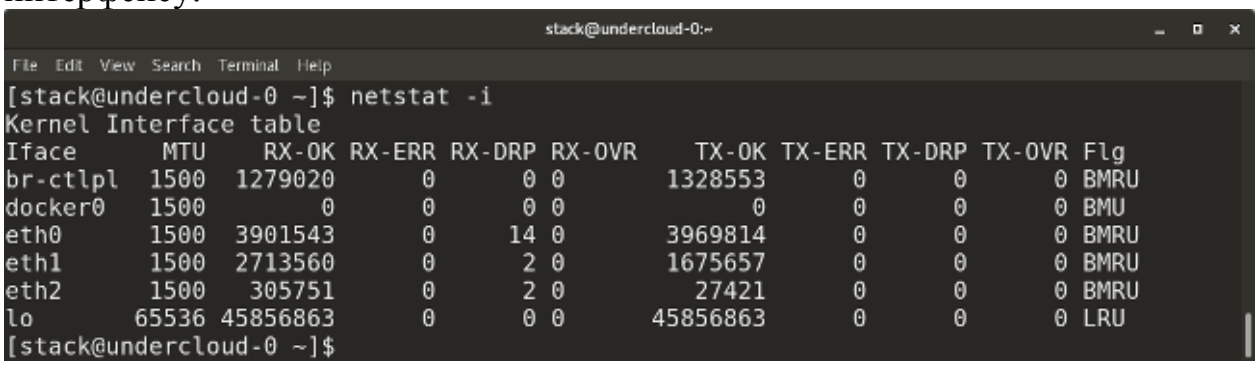
Вот вариант этой команды, выводящий статистику по протоколам:

```
netstat -s
```



Статистика по протоколам

Следующий вариант вызова *netstat* позволяет узнать сведения об отправленных и полученных пакетах (*transmission/receive, TX/RX*) по каждому интерфейсу:



Данные об отправленных и полученных пакетах

Nmcli

Утилита *nmcli* отлично подходит для управления сетевыми соединениями, для выполнения настроек и для решения других подобных задач. С её помощью можно управлять программой *NetworkManager* и модифицировать сетевые параметры различных устройств.

Примеры

Вот как с помощью *nmcli* вывести список сетевых интерфейсов:

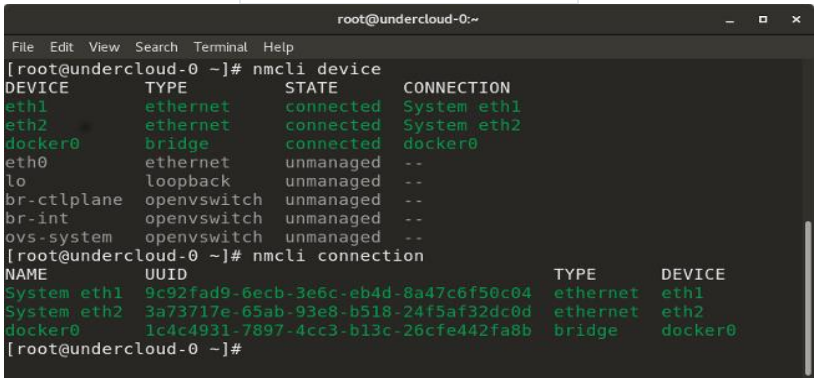
nmcli device

Так можно вывести информацию по конкретному интерфейсу:

nmcli device show <interface>

Следующий вариант вызова команды позволяет проверить подключение устройства к сети:

nmcli connection



Примеры использования *nmcli*

Эта команда позволяет отключить заданный интерфейс:

nmcli connection down <interface>

А эта позволяет включить интерфейс:

nmcli connection up <interface>

Вот пример команды, которая добавляет *VLAN*-интерфейс с заданным *VLAN*-номером, *IP*-адресом и шлюзом к указанному интерфейсу:

nmcli con add type vlan con-name <connection-name> dev <interface> id <vlan-number> ipv4 <ip/cidr> gw4 <gateway-ip>

Маршрутизация

Существует множество команд, которые можно использовать для проверки правил маршрутизации и их настройки. Рассмотрим самые полезные из них.

Примеры

Следующая команда показывает все текущие маршруты, настроенные для соответствующих интерфейсов:

ip route

Маршруты, настроенные для интерфейсов

Эта команда позволяет добавить в таблицу маршрутизации шлюз, используемый по умолчанию:

route add default gw <gateway-ip>

Следующая команда добавляет в таблицу маршрутизации новый сетевой маршрут. Существует и множество других её параметров, позволяющих выполнять такие операции, как добавление маршрута и шлюза, используемых по

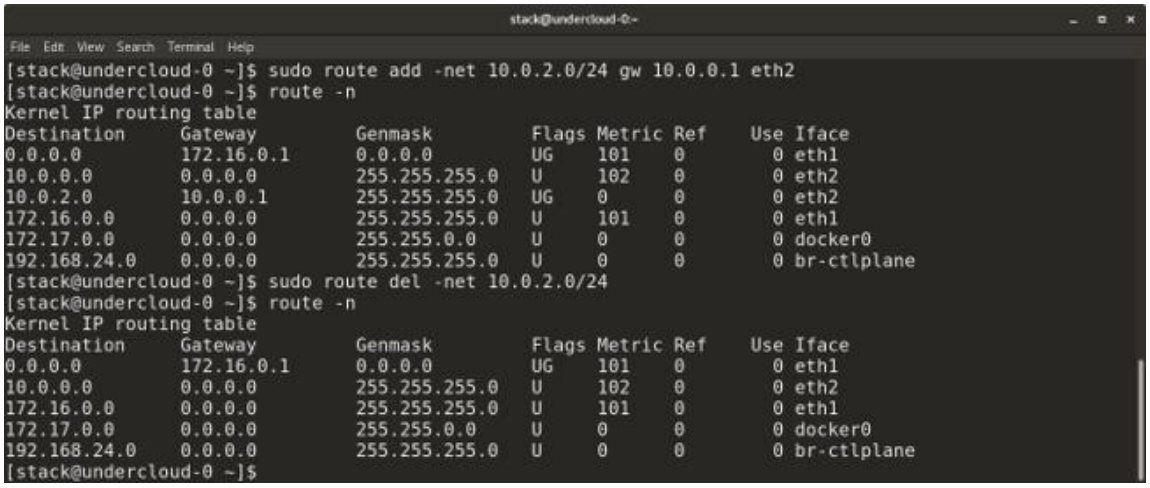
умолчанию, и так далее.

route add -net <network ip/cidr> gw <gateway ip> <interface>

С помощью такой команды можно удалить запись о заданном маршруте из таблицы маршрутизации:

route del -net <network ip/cidr>

Примеры использования команды *route*.

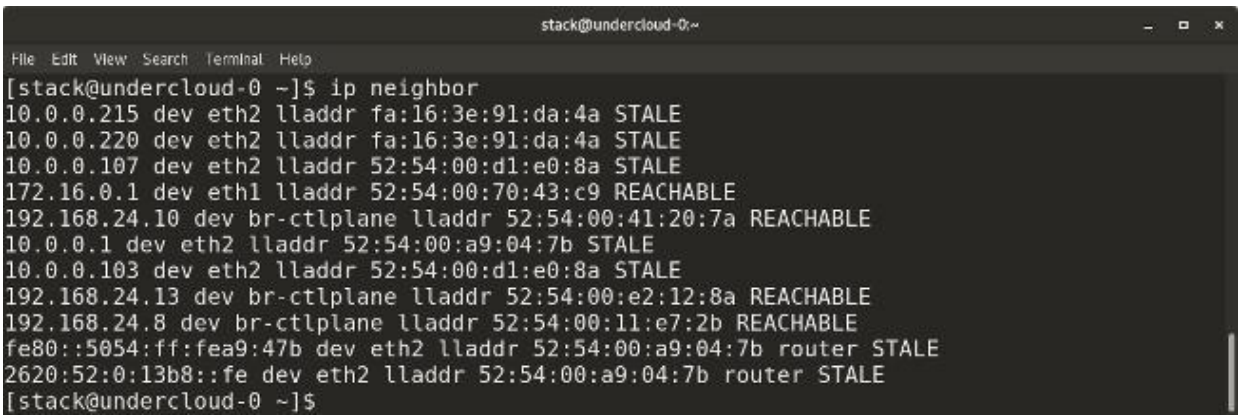


Использование команды *route*

Вот команда, которая применяется для вывода текущей таблицы соседей. Кроме того, её можно использовать для добавления, изменения или удаления сведений о соседях:

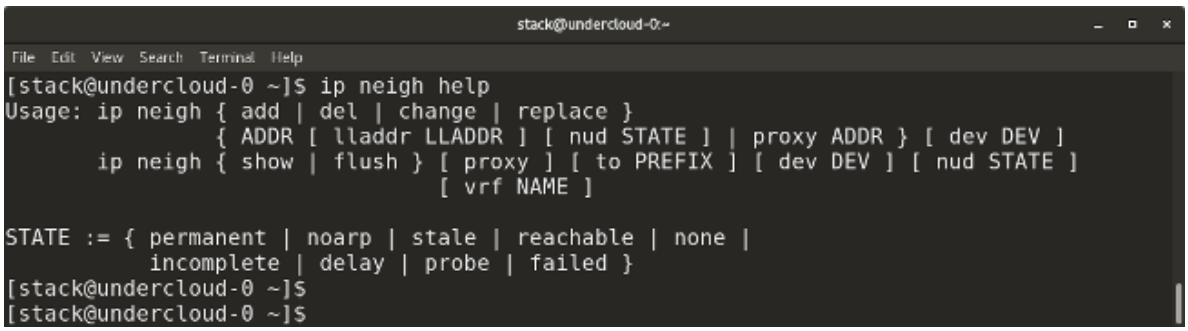
ip neighbor

Примеры её использования.



Данные, полученные с помощью команды *ip neighbor*

Вот сведения о команде *ip neigh*

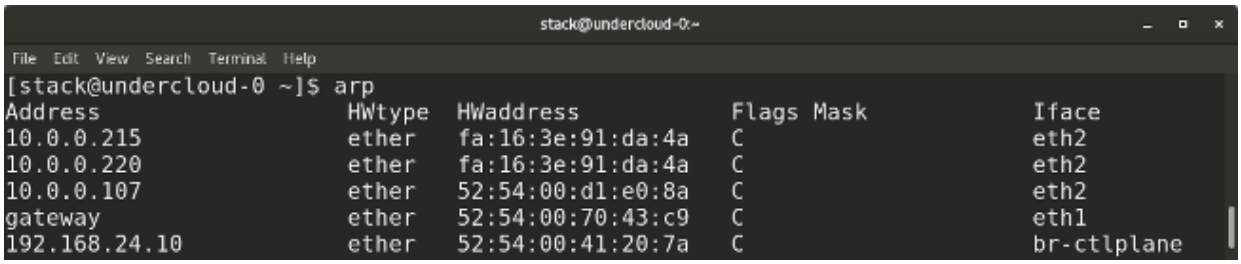


Сведения о команде *ip neigh*

Команда *arp* (*ARP* – это сокращение от *Address Resolution Protocol*, протокол определения адреса) похожа на *ip neighbor*. Утилита *arp* выводит данные о соответствии *IP*-адресов *MAC* -адресам. Вот как её использовать:

arp

Пример её вызова.



Вызов команды *arp*

Tcpdump и *Wireshark*

Linux даёт в распоряжение администратора множество инструментов для захвата и анализа пакетов. Среди них, например, tcpdump, wireshark, tshark и другие. Они используются для захвата сетевого трафика в передаваемых системой пакетах или в пакетах, получаемых ей. Это делает их весьма ценным инструментом администратора, помогающим в деле выяснения причин различных сетевых неполадок. Тем, кто предпочитает командную строку всем остальным способам общения с компьютерами, понравится tcpdump. Тем же, кто любит графические интерфейсы, можно порекомендовать wireshark – отличный инструмент для захвата и анализа пакетов. Утилита tcpdump – это встроенное в Linux средство для захвата сетевого трафика. Его можно использовать для захвата и вывода трафика с фильтрацией по портам, протоколам, и по другим признакам.

Примеры

Такая команда показывает, в режиме реального времени, пакеты с заданного

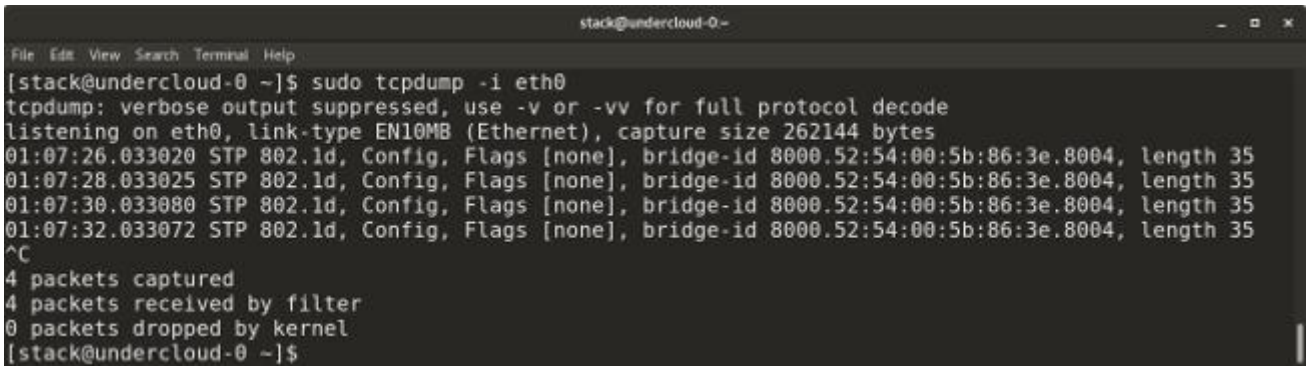
интерфейса:

tcpdump -i <interface-name>

Пакеты можно сохранять в файл, воспользовавшись флагом *-w* и задав имя файла:

tcpdump -w <output-file.> -i <interface-name>

Пример использования *tcpdump*.



Использование *tcpdump*

Следующий вариант команды используется для захвата пакетов, приходящих с заданного *IP* системы-источника:

tcpdump -i <interface> src <source-ip>

Так можно захватить пакеты, идущие на заданный адрес системы-приёмника:

tcpdump -i <interface> dst <destination-ip>

Пример использования *tcpdump* для захвата пакетов для заданного номера порта, например, это может быть порт 53, 80, 8080, и так далее:

tcpdump -i <interface> port <port-number>

Здесь показано, как с помощью tcpdump захватывать пакеты заданного протокола, вроде TCP, UDP или других:

tcpdump -i <interface> <protocol>

Iptables

Утилита *iptables* похожа на файрвол, она поддерживает фильтрацию пакетов, что позволяет управлять трафиком, пропуская или блокируя его. Диапазон возможностей этой утилиты огромен. Рассмотрим несколько наиболее распространённых вариантов её использования.

Примеры

Следующая команда позволяет вывести все существующие правила *iptables*:

```
iptables -L
```

Эта команда удаляет все существующие правила:

```
iptables -F
```

Следующие команды разрешают прохождение трафика с заданного номера порта к заданному интерфейсу:

```
iptables -A INPUT -i <interface> -p tcp –dport <port-number> -m state –state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o <interface> -p tcp -sport <port-number> -m state – state
ESTABLISHED -j ACCEPT
```

Следующие команды разрешают *loopback*-доступ к системе:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Nslookup

Инструмент *nslookup* используется для получения сведений о назначении *IP*-адресов сетевым ресурсам. Его можно использовать и для получения сведений с *DNS*-серверов, например, таких, как все *DNS*-записи для некоего веб-сайта (ниже мы рассмотрим соответствующий пример). На *nslookup* похожа утилита *dig* (*Domain Information Groper*).

Примеры

Следующая команда выводит *IP*-адреса вашего *DNS*-сервера в поле *Server*, и, ниже, выдаёт *IP*-адрес искомого сайта:

nslookup <website-name.com>

Такая команда показывает все доступные записи для заданного веб-сайта или домена:

nslookup -type=any <website-name.com>

Поиск неполадок

Вот набор команд и список важных файлов, используемых для идентификации сетевых неполадок.

Примеры

- *Ss* – утилита для вывода статистической информации о сокетах.
- *nmap <ip-address>* – имя этой команды является сокращением от *Network Mapper*. Она сканирует сетевые порты, обнаруживает хосты, выясняет *MAC*-адреса и выполняет множество других задач.
- *ip addr/ifconfig -a* – эта команда предоставляет сведения об *IP*-адресах и другие данные по всем интерфейсам системы.
- *ssh -vvv user@<ip/domain>* – такая команда позволяет подключиться по *SSH* к другому компьютеру, используя заданный *IP*-адрес или доменное имя компьютера и имя пользователя. Флаг *-vvv* позволяет получать подробные сведения о происходящем.
- *ethtool -S <interface>* – данная команда позволяет вывести статистические сведения по заданному интерфейсу.
- *ifup <interface>* – эта команда включает указанный интерфейс.
- *ifdown <interface>* – эта команда отключает указанный интерфейс.
- *systemctl restart network* – с помощью этой команды можно перезагрузить системную сетевую подсистему.
- */etc/sysconfig/network-scripts/<interface-name>* – это файл настройки интерфейсов, используемый для указания *IP*-адреса, сети, шлюза и других параметров для заданного интерфейса. Здесь можно задать использование интерфейсом *DHCP*-режима.
- */etc/hosts* – данный файл содержит сведения о соответствии хостов или доменов *IP*-адресам, настроенные администратором.
- */etc/resolv.conf* – в этом файле хранятся настройки *DNS*.
- */etc/ntp.conf* – этот файл хранит настройки *NTP*.

8. Управление учётными записями пользователей и доступом к ресурсам

Управление *учётными записями пользователей* и *группами* – важная часть работы системного администратора в организации. Но что бы делать её эффективно, хороший системный администратор должен прежде всего понимать, что такое учётные записи пользователей и группы, и как они работают.

Основное предназначение учётных записей пользователей – идентифицировать человека, использующего компьютерную систему. Второе (но также важное) предназначение учётных записей пользователей – назначение каждому человеку ресурсов и прав доступа.

Ресурсами могут быть файлы, каталоги и устройства. Управление доступом к этим ресурсам составляет большую часть будничной работы системного администратора, и часто доступ к ресурсам организуется с помощью групп. Группы – это логические конструкции, с помощью которых можно объединить учётные записи пользователей для какой-то общей цели. Например, если в организации несколько системных администраторов, всех их можно поместить в одну группу системных администраторов. Затем этой группе можно дать разрешения на доступ к ключевым ресурсам системы. Таким образом, группы могут быть мощным инструментом управления ресурсами и доступом.

Учётные записи пользователей

Как было сказано ранее, учётные записи представляют собой средства идентификации пользователей и проверки их подлинности. Учётные записи пользователей имеют несколько компонентов. Первый компонент – имя пользователя. Второй – пароль, а затем идёт информация об управлении доступом.

Имя пользователя

С точки зрения системы, имя пользователя является ответом на вопрос: «Кто вы?». А значит, главное требование, связанное с именами пользователей – они должны быть уникальными. Другими словами, имя каждого пользователя должно отличаться имён всех остальных пользователей данной системы.

Вследствие этого требования, крайне важно определить на будущее, как будут выбираться имена пользователей. В противном случае вы можете оказаться в положении, когда вам придётся что-то придумывать для каждого нового пользователя, которому нужна учётная запись.

Что вам нужно, так это соглашение об именовании учётных записей ваших пользователей.

Соглашения об именовании

Выбрав для себя соглашение об именах пользователей, вы можете избежать многих проблем. Вместо того, чтобы сочинять имена, каждый раз, как они понадобятся (при этом придумать подходящее имя будет всё труднее и труднее),

вы думаете об этом заранее и разрабатываете соглашение для всех создаваемых впоследствии учётных записей. Ваше соглашение об именовании может быть очень простым, а может быть его описание займёт несколько страниц.

Каким именно будет ваше соглашение об именах, должно зависеть от нескольких факторов:

- размера организации;
- структуры организации;
- природы организации.

Размер вашей организации имеет значение, так как он определяет, для скольких пользователей должно быть разработано ваше соглашение об именах. Например, в очень маленькой организации, возможно, все могут использовать своё имя. Но для больших организаций это соглашение об именах не подойдёт.

Структура организации также может накладывать отпечаток на соглашение об именовании. Для организаций с чётко определённой структурой, возможно, будет уместно включить в соглашение об именовании элементы этой структуры. Например, вы можете включить в имя пользователя коды отделов вашей организации.

Кроме этого, одни соглашения об именовании могут быть более подходящими, чем другие, в зависимости от общей сущности вашей организации. Организация, которая имеет дело с сильно засекреченными данными, может выбрать соглашение, не позволяющее связать имя учётной записи пользователя с его настоящим именем.

Ниже перечислены некоторые соглашения об именовании, которые используются в обычных организациях:

- Имя (*robert, indiana, eva* и т.д.).
- Фамилия (*smith, jones, brown* и т. д.).
- Первая буква имени и фамилия (*rsmith, ijones, ebrown* и т. д.).
- Фамилия и код отдела (*smith*029, *jones*45*4, brown*191 и т. д.).

В описанных здесь соглашениях об именовании есть один общий недостаток – есть вероятность, что когда-нибудь появятся два человека, которым, следуя выбранному соглашению, нужно будет дать одинаковое имя пользователя. Это называется *конфликтом* имён. Так как все имена пользователей должны быть уникальны, проблему конфликта имён необходимо решить. Следующий раздел рассказывает, как это сделать.

Разрешение конфликта имён

Конфликты имён в жизни неизбежны – как бы вы ни пытались, рано или поздно вам придётся с ними столкнуться. Вы должны учесть это в своём соглашении об именовании. Это можно сделать разными способами:

- Добавлять в конфликтующие имена последовательные номера (*smith, smith*1, *smith*2 и т.д.).
- Добавлять в конфликтующие имена какие-то атрибуты пользователя (*smith, rsmith, rosmith* и т. д.).

- Добавлять в конфликтующие имена данные организационной структуры (*smith, smith*029, *smith*454 и т.д.).

Любое соглашение об именовании обязательно должно описывать порядок разрешения конфликтов. Однако для людей извне организации это сильно затрудняет точное определение имени пользователя. Поэтому недостатком большинства соглашений об именовании является возможность ошибки в имени получателя почты.

Процедура изменения имени

Если в вашей организации применяется соглашение об именовании, основанное на имени пользователей, вам неизбежно придётся столкнуться с необходимостью изменить имя. Даже если не меняется настоящее имя человека, время от времени требуется внести изменения в имя пользователя. Причины могут быть самыми разными: пользователю не нравится имя его учётной записи, или он занимает высокое положение в вашей организации и, используя его, желает получить «подобающее» имя пользователя.

Какова бы ни была причина, изменяя имя пользователя нужно помнить о нескольких моментах:

- Внести изменения на *всех* затронутых системах.
- Сохранить идентификаторы пользователя на нижнем уровне без изменений.
- Сменить владение всеми файлами или другими ресурсами пользователя (если это необходимо).
- Решить проблемы, связанные с электронной почтой.

Первое и самое главное, важно распространить изменения имени на все системы, в которых до этого использовалось старое имя. В противном случае, функции операционной системы, зависящие от имени пользователя, могут в одних системах работать, а в других – нет. В некоторых операционных системах средства управления доступом привязаны к именам пользователей, в таких системах проблемы, связанные с изменением имени, особенно актуальны.

Во многих операционных системах в большинстве функций, связанных с пользователями, используется некоторый код, идентифицирующий пользователя. Чтобы сократить число проблем, вызванных изменением имени пользователя, старайтесь сохранить идентификационный код от старого имени пользователя. Если код не сохраняется, часто возникает ситуация, когда пользователь больше не может обращаться к файлам и другим ресурсам, которыми он владел до этого с прежним именем пользователя.

Если избежать смены кода пользователя нельзя, необходимо также обновить данные владения для всех файлов и других ресурсов пользователя, чтобы на них отразилось это изменение. Выполнить это без ошибок сложно, потому что всегда в каком-нибудь забытом углу системы что-то остаётся без внимания.

Пожалуй, сложнее всего поменять имя, когда это касается электронной почты. Это объясняется тем, что, если не были предприняты специальные меры,

почта, отправленная по старому имени пользователя, не будет доставлена по новому имени.

К сожалению, проблемы, связанные с изменением имени пользователя, в контексте электронной почте приобретают многоплановый характер. По сути, изменение имени пользователя означает, что люди больше не будут знать правильное имя пользователя этого человека. На первый взгляд, это не кажется большой проблемой — нужно всего лишь сообщить об изменении всем сотрудникам вашей организации. Но как быть с теми, кто посылает этому человеку почту, но не работает в вашей организации? Как сообщить об изменении? И что делать со списками рассылки (внутренними и внешними)? Как обновить их?

Простого ответа на эти вопросы нет. Возможно, лучше всего будет создать псевдоним почтового ящика, чтобы вся почта, отправляемая по старому имени, автоматически перенаправлялась в ящик с новым именем. Затем можно попросить пользователя сообщить об изменении имени всем, с кем он ведёт переписку. Со временем по адресу псевдонима будет приходить всё меньше и меньше писем, и в конце концов этот псевдоним можно будет удалить.

Хотя псевдонимы, в некоторой степени, увековечивают неверное предположение (что пользователь, сейчас известный под именем *esmith*, по-прежнему известен под именем *ejones*) – это единственный способ гарантировать, что почта будет доходить до нужного человека.

Пароли

Если имя пользователя даёт ответ на вопрос: «Кто вы?», пароль — ответ на неизбежно следующее за этим требованием: «Докажите это!»

Говоря более формально, пароль даёт возможность подтвердить подлинность человека, заявляющего, что он является пользователем с заданным именем. Эффективность схемы проверки подлинности по паролю зависит от нескольких свойств пароля:

- Секретность пароля.
- Устойчивость пароля к угадыванию.
- Устойчивость пароль к атаке перебором.

Пароли, обладающие всеми этими свойствами, называются *сильными*, а не обладающие одним из этих свойств – *слабыми*. Создание сильных паролей имеет большое значение для безопасности организации, так как узнать или угадать сильный пароль гораздо сложнее. Обеспечить использование только сильных паролей можно двумя способами:

- Системный администратор может сам назначать пароли всем пользователям.
- Системный администратор может разрешить пользователям задавать пароли самостоятельно, но проверять, достаточно ли сильны эти пароли.

Назначение паролей для всех пользователей гарантирует, что пароли будут сильными, но это становится непосильной задачей по мере роста организации. При

этом также увеличивается опасность того, что пользователи начнут записывать свои пароли.

Поэтому многие системные администраторы предпочитают, чтобы пользователи назначали себе пароли сами. Тем не менее, хороший системный администратор принимает меры для проверки, насколько сильны пароли.

Каждый системный администратор должен в полной мере осознавать необходимость сохранения паролей в секрете. Однако об этом совершенно не думают многие пользователи. На самом деле, многие пользователи вообще не видят различий между именами пользователей и паролями. Учитывая этот печальный факт, важно провести с пользователями разъяснительную работу, чтобы пользователи понимали, что пароль нужно хранить в секрете, так же, как и банковскую карту.

Пароли должны быть сложны, чтобы угадать их было невозможно. Сильным паролем считается пароль, угадать который злоумышленник не сможет, даже если он хорошо знает пользователя.

Атака на пароль методом перебора подразумевает систематический перебор (обычно с помощью программы, называемой *взломщиком паролей*) всех возможных комбинаций символов в надежде на то, что в конце концов нужный пароль будет найден. Сильный пароль должен быть образован таким образом, чтобы число потенциальных паролей, которые требовалось бы проверить, было очень большим, и на поиск пароля уходило бы очень много времени.

Слабые пароли

Как было сказано ранее, слабый пароль не проходит одной из следующих проверок:

- Он секретный
- Он устойчив к угадыванию
- Он устойчив к атаке перебором.

Короткие пароли

Короткий пароль является слабым, так как он гораздо больше подвержен атаке перебором. Чтобы это проиллюстрировать, рассмотрим табл.1, в которой показано число паролей, которые потребуется протестировать в атаке методом перебора. (Подразумевается, что пароли состоят только из букв в нижнем регистре).

Длина пароля и число вариантов паролей Таблица 1.

Длина пароля	Число вариантов паролей
1	26
2	676
3	17576

4	456 976
5	11 881 376
6	308 915 776

Ограниченный набор символов

Число различных символов, которые могут составлять пароль, в значительной мере влияет на возможность успешного проведения атаки перебором. Например, что будет, если к 26 разным буквам в нижнем регистре, которые мы могли использовать в пароле, мы добавим ещё и цифры? Это будет означать, что каждый символ в пароле может выбираться не из 26, а из 36 символов. В случае с паролем из шести символов, это увеличивает число возможных паролей с 308 915 776 до 2 176 782 336.

Но и это ещё не всё. Если мы рассмотрим алфавитно-цифровые пароли смешанного регистра (для тех операционных систем, которые это поддерживают), число возможных паролей из шести символов вырастет до 56 800 235 584. Добавление других символов (например, знаков препинания) ещё больше увеличивает число возможных паролей, что многократно усложняют атаку перебором.

Однако, надо помнить о том, что атака перебором – не единственный вид атаки для получения пароля. Следующие разделы рассказывают о других вариантах слабых паролей.

Известные слова

Многие атаки на пароли основаны на том факте, что людям больше всего нравятся пароли, которые они могут вспомнить. И для большинства людей запоминаемые пароли – те, что содержат слова. Поэтому большинство атака выполняется с применением словаря. Другими словами, взломщик использует списки слов, пытаясь найти слово или набор слов, составляющих пароль.

Личные сведения

Пароли, содержащие личные сведения (имя или день рождения любимой, имя своего питомца или личный номер), возможно, и не будут раскрыты с помощью атаки по словарю. Однако, если злоумышленник знает вас лично (или настолько заинтересован в пароле, что изучил вашу личную жизнь), он сможет угадать ваш пароль практически без труда.

Помимо словарей, многие программы для взлома паролей также перебирают распространённые имена, даты и другие подобные сведения. Поэтому, даже если взломщик не знает, что вашу собаку зовут *Gracie*, с помощью хорошей программы взлома паролей они смогут узнать, что ваш пароль «*mydogisgracie*».

Простые фокусы со словами

Использование в качестве основы для пароля каких-то из упомянутых выше сведений, но с перестановкой символов наоборот, не сделать слабый пароль сильным. Большинство программ взлома пароля делают такие же фокусы с возможными паролями. В число таких фокусов входит замена определённых букв цифрами в распространённых словах. Вот некоторые примеры:

- *drowssaPdaB1;*
- *R3allyP00r.*

Один пароль в разных системах

Даже если у вас есть сильный пароль, использовать его в нескольких системах – плохая идея. Очевидно, что это вряд ли возможно, если все системы используют центральный сервер проверки подлинности, но в остальных случаях во всех системах следует использовать разные пароли.

Пароли, записанные на бумаге

Ещё один способ сделать сильный пароль слабым – записать его на бумаге. Если вы запишите его на бумаге, у вас больше не будет проблемы секретности, у вас будет проблема безопасности – теперь вы должны хранить этот листок бумаги в безопасном месте. Следовательно, записывать пароли на бумаге – очень плохая идея.

Однако в некоторых организациях правила требует записывать пароли. Например, некоторые организации хранят записанные пароли на случай потери ключевых сотрудников (например, системных администраторов). В таких случаях бумага с паролями хранится в физически защищённом месте, попасть в которое смогут только несколько определённых человек вместе. Часто для этого используются камеры хранения с несколькими замками и депозитные ящики в банке.

Любая организация, хранящая таким образом пароли на случай аварийной ситуации, должны понимать, что существование записанных на бумаге пароле привносит элемент риска в безопасности их систем, вне зависимости от того, насколько безопасно хранятся записанные пароли. Это особенно справедливо, когда хорошо известно, что пароли записываются на бумаге (и где они хранятся).

К сожалению, довольно часто встречаются записанные пароли, которые не входят в план восстановления и не хранятся в сейфах – это пароли обычных пользователей, которые хранятся в следующих местах:

- В ящике письменного стола (закрывающемся или нет).
- Под клавиатурой.
- В бумажнике.
- На бумаге, приклеенной сбоку к монитору.

Ни одно из этих мест совершенно не годится для записанного пароля.

Сильные пароли

Мы увидели, какие пароли являются слабыми, в следующих разделах обсуждаются свойства, которые характерны для всех сильных паролей.

Длинные пароли

Чем длиннее пароли, тем меньше вероятность того, что атака перебором закончится успешно. Поэтому если это поддерживает ваша операционная система, назначьте для своих пользователей довольно большую минимально допустимую длину пароля.

Расширенный набор символов

Постарайтесь добиться использования алфавитно-цифровых паролей со смешанным регистром, и настоятельно требуйте включать во все пароли как минимум один не алфавитно-цифровой символ:

- *t1Te-Bf,te*
- *Lb@lbhom*

Запоминаемые пароли

Пароль является сильным, только если его можно запомнить. Однако обычно запоминаемый пароль и легко угадываемый – почти синонимы. Поэтому дайте своим пользователям несколько советов, как следуют выбирать запоминаемые пароли, угадать которые будет не так просто.

Например, можно взять любимое изречение или какую-то фразу, и выбрать первые буквы каждого слова как основу для нового пароля. Результат будет запоминаемым (потому что фраза, из которой он получен, сама запоминаемая), но при этом он не будет содержать слов.

Срок действия пароля

Если это вообще возможно, внедрите в своей организации ограничение срока действия пароля. Возможность ограничивать срок действия пароля (имеющаяся во многих операционных системах) накладывает ограничения на время, в течение которого пароль считается верным. В конце срока жизни пароля пользователю предлагается ввести новый пароль, который затем может быть использоваться, пока срок его жизни тоже не закончится.

Главный вопрос по поводу срока действия пароля, который встаёт перед многими системными администраторами, это вопрос выбора этого срока. Какими он должен быть?

При выборе срока жизни пароля есть два диаметрально противоположных направления:

- Удобство пользователя.
- Безопасность.

В одной крайней точке – при сроке действия пароля 99 лет у пользователя будет очень мало неудобств (если они вообще будут). Однако это также почти не улучшит безопасность (если вообще улучшит).

В другой крайней точке – при сроке действия пароля 99 минут пользователи испытают большие неудобства. Однако безопасность будет значительно улучшена.

Поэтому нужно найти баланс между интересами ваших пользователей, которым нужно удобство, и вашей организации, которой нужна безопасность. В большинстве организаций чаще всего используется срок жизни паролей, измеряемый неделями или месяцами.

Информация об управлении доступом

Помимо имени и пароля пользователя, учётные записи также содержат информации об управлении доступом. Эта информация принимает разные формы в зависимости от используемой операционной системы. Но чаще всего это следующая информация:

- Информация уровня системы о пользователе.
- Информация уровня системы о группах.
- Список дополнительных групп/сущностей, членом которых является пользователь.
- Информация о доступе по умолчанию, применяемая ко всем создаваемым пользователем файлам и ресурсам.

В некоторых организациях, информация об управлении доступом может никогда не меняться. Чаще всего это имеет место на отдельных, личных рабочих станциях. В других же организациях, особенно в тех, где различные группы пользователей интенсивно используют общий доступ к сетевым ресурсам, информация об управлении доступом меняется очень часто.

Нагрузка, необходимая для поддержания информации об управлении доступом ваших пользователей, зависит от того, насколько интенсивно в вашей организации используются возможности управления доступом вашей операционной системы. Хотя использовать эти возможности вовсе не плохо (и на самом деле это может быть обязательно), это значит, что окружение вашей системы может потребовать больше усилий для поддержки и появится больше способов ошибиться в настройках каждой учётной записи.

Поэтому, если это необходимо в вашей организации, вы должны придать большое значение точному описанию действий, необходимых для создания и правильной настройки учётной записи. На самом деле, если существует несколько типов учётных записей пользователей, вы должны документировать каждую (создание новой учётной записи для нового пользователя финансового отдела, операционного отдела и т. д.).

Повседневное управление учётными записями и доступом к ресурсам

Как говорит старая мудрость, «Постоянны только изменения». И это также касается сообщества ваших пользователей. Одни люди приходят, другие уходят, а третьи переходят с одной должности на другую. Поэтому системные администраторы должны быть способны реагировать на изменения, которые являются обычными явлениями в повседневной работе вашей организации.

Новые сотрудники

Когда в организацию приходит новый человек, обычно ему разрешается доступ к различным ресурсам (в зависимости от его должности). Ему может быть выделено рабочее место, телефон и ключ от входной двери.

Ему также может быть разрешён доступ к одному или нескольким компьютерам в вашей организации. Ваша задача, как системного администратора, проконтролировать, что всё сделано правильно и своевременно. Как же решить эту задачу?

Прежде чем вы что-нибудь сделаете, вас должны уведомить о приходе нового сотрудника. В разных организациях это делается по-разному. Например, возможны следующие варианты:

- Разработать процедуру, согласно которой о приходе нового сотрудника вас уведомляет отдел кадров вашей организации.
- Создать форму, которую будет заполнить начальник этого сотрудника и передавать вам для создания учётной записи.

Разным организациям нужны разные подходы. Как бы это ни было организовано, крайне важно, чтобы у вас был чёткий порядок уведомлений о необходимости выполнения любых действий, связанных с учётными записями.

Прекращение работы

Уход людей из организации – неизбежное явление. Иногда это происходит при благоприятных обстоятельствах, а иногда – нет. В любом случае важно, чтобы вы предусмотрели эту ситуацию и могли предпринять соответствующие действия.

По меньшей мере, это должны быть следующие действия:

- Запрет доступа пользователей ко всем системам и связанным ресурсам (обычно для этого меняется или блокируется пароль пользователя).
- Копирование в архив файлов пользователя, если они содержат что-то, что может понадобится позже.
- Координация доступа к файлам начальником пользователя.

Самое главное – защитить ваши системы от пользователя, прекратившего работу. Это особенно важно, если пользователь покинул организацию при таких обстоятельствах, при которых у него могло остаться недоброжелательное отношение к вашей организации. Но даже если обстоятельства более благоприятные, в интересах вашей организации, чтобы вы быстро и надёжно запретили доступ пользователю, прекратившему работу.

Это показывает, что вас должны уведомлять обо всех подобных событиях, и предпочтительнее до того, как сотрудник прекратит работу фактически. Чтобы получать информацию о предстоящих увольнениях заранее, вам следует сотрудничать с отделом кадров вашей организации.

После того, как вы запретите доступ, сделайте копию файлов сотрудника, покидающего организацию. Эта копия может быть сделана в рамках стандартного резервного копирования, или в рамках специальной процедуры копирования в архив данных старых учётных записей. В определении наиболее подходящего способа выполнения резервной копии играют роль разные факторы, в частности,

регламент сохранения данных, требования сохранения доказательств на случай судебных разбирательств и т. д.

В любом случае на данном этапе рекомендуется сделать резервную копию, так как на следующем этапе (предоставление начальнику доступа к файлам ушедшего сотрудника) они могут быть случайно удалены. В таких обстоятельствах, имея текущую копию, можно легко восстановить данные, что упрощает этот процесс и для начальника, и для вас.

Теперь вы должны определить, какой доступ к файлам ушедшего сотрудника нужен его начальнику. В зависимости от вашей организации и рода занятий ушедшего сотрудника, возможно, доступ не следует открывать вовсе, или, наоборот, разрешить доступ ко всему, что потребуется начальнику.

Если этот человек не только от случая к случаю пользовался почтой, скорее всего его начальнику придётся отсортировать файлы и определить, что следует оставить, а что может быть удалено. По завершению этого процесса как минимум некоторые файлы можно будет передать сотруднику или сотрудникам, которым перешли обязанности ушедшего сотрудника. Возможно, на этом, последнем этапе всего процесса потребуется ваша помощь, или начальник сможет справиться с этим сам. Это зависит от файлов и сущности работы, которую выполняет ваша организация.

Переход на другую должность

Удовлетворение запросов на создание учётных записей для новых пользователей и отработка последовательности событий для блокировки учётной записи при уходе сотрудника – достаточно простые процессы. Однако всё не так однозначно, когда сотрудник организации переходит на другую должность. Иногда при этом может потребоваться внести изменения в учётную запись, а иногда нет.

Есть как минимум три человека, которые должны обеспечить подходящую перенастройку учётную записи в соответствии с новой должностью:

- Вы.
- Предыдущий начальник пользователя.
- Новый начальник пользователя.

Между собой вы должны определить, что нужно сделать, чтобы полностью закрыть старые задачи пользователя, и что нужно сделать, чтобы подготовить учётную запись пользователю к новому кругу задач. Во многих отношениях этот процесс можно представить, как отключение существующей учётной записи пользователя и создание новой. На самом деле в некоторых организациях именно так и происходят переводы сотрудников с одной должности на другую.

Однако, скорее всего, учётная запись пользователя будет сохранена и изменена в соответствии с его новым кругом задач. Этот подход означает, что вы должны внимательно проверить учётную запись, чтобы убедиться в том, что ему доступны только те ресурсы и назначены только те привилегии, что необходимы для новых задач.

Ещё больше усложняет эту ситуацию тот факт, что часто в процессе перехода пользователь временно выполняет задачи, связанные с обеими должностями. Здесь

вы можете обратиться к предыдущему и новому начальникам, чтобы они определили длительность этого переходного периода.

Управление ресурсами пользователей

Создание учётных записей пользователей – это только часть работы системного администратора. Не менее важная задача – управление ресурсами пользователей. Поэтому необходимо рассмотреть следующие вопросы:

- Кому разрешён доступ к общим данным?
- Куда пользователи обращаются за этими данными?
- Каким образом предотвращается нецелевое использование ресурсов?

Кому разрешён доступ к общим данным?

Доступ пользователя к конкретному приложению, файлу или каталогу определяется разрешениями, заданными для этого ресурса.

Кроме того, часто полезно иметь возможность назначать разные разрешения разным классам пользователей. Например, в общем временном хранилище стоит предотвратить случайное (или злонамеренное) удаление файлов пользователя другими пользователями, и при этом разрешить владельцу файла общий доступ.

Другой пример – доступ к домашнему каталогу пользователя. Создание или просмотр файлов в домашнем каталоге должен быть разрешён только его владельцу. Другим пользователям доступ должен быть запрещён (если только владелец каталога не захочет обратного). Это помогает сохранять данные пользователя в секрете и предотвращает хищение личных файлов.

Но очень часто бывает, что нескольким пользователям требуется доступ к одному ресурсу компьютера. В таких случаях может возникнуть необходимость в создании соответствующих общих групп.

Общие группы и данные

Как было сказано во введении, группы – это логические конструкции, с помощью которых можно объединить учётные записи пользователей для какой-то определённой цели.

Управляя пользователями в организации, рекомендуется определить, какие данные должны быть доступны конкретным отделам, какие данные должны быть закрыты для других и какие данные должны быть доступны всем. Это поможет создать подходящую структуру групп и соответствующим образом распределить доступ к общим данным.

Например, предположим, что отдел, обрабатывающий поступающие счета, должен вести список счётов, не оплаченных вовремя. Этот список также должен быть доступен отделу по работе с задолженностью. Если сотрудники и отдела обработки поступающих счетов, и отдела по работе с задолженностью являются членами группы *accounts*, этот список можно поместить в общий каталог

(владельцем которого будет группа *accounts*) и разрешить в нём этой группе чтение и запись.

Определение структуры групп

Когда системные администраторы начинают создавать общие группы, перед ними встают разные вопросы, в частности:

- Какие группы создавать?
- Кого включить в конкретную группу?
- Какие разрешения должны быть назначены для заданных общих ресурсов?

Ответить на эти вопросы поможет здравый смысл. Во-первых, создавая группы можно отразить структуру вашей организации. Например, если у вас есть финансовый отдел, создайте группу *finance* и включите в эту группу всех сотрудников данного отдела. Если финансовая информация не должна быть доступна всей компании, но руководство организации должно иметь к ней доступ, разрешите ему доступ к файлам и каталогам финансового отдела, добавив всех руководителей в группу *finance*.

Разрешая пользователям доступ, следует проявлять осмотрительность, чтобы важная информация случайно не попала в чужие руки.

Подходя к созданию структуры групп организации таким образом, можно безопасно и эффективно удовлетворить требования к предоставлению совместного доступа к данным.

Куда пользователи обращаются за общими данными?

Для организации доступа пользователей к общим данным обычно используется центральный сервер или группа серверов, которые предоставляют определённые каталоги другим компьютерам в сети. Это позволяет хранить данные в одном месте, синхронизация данных между разными компьютерами не нужна.

Прежде чем внедрить это на практике, вы должны определить, какие компьютеры будут обращаться к централизованным данным. Для этого следует учесть, какие операционные системы работают на ваших компьютерах. Это может повлиять на практическую реализацию данного подхода, так как ваш сервер хранения должен быть способен предоставлять данные всем операционным системам, используемым в вашей организации.

К сожалению, если данные совместно используется несколькими компьютерами в сети, это может приводить к конфликтам владения файлами.

Вопросы глобального владения

Централизованное хранение данных и доступ к ним по сети нескольких компьютеров имеет свои преимущества. Однако, предположим на минуту, что на каждом из этих компьютеров есть локальный список учётных записей пользователей. Как всё это будет работать, если список пользователей на отдельных компьютерах не будет совпадать со списком пользователей на центральном сервере? Или эти списки пользователей на разных компьютерах вообще не будут соответствовать друг другу?

Во многом это зависит от того, как в каждой системе реализованы пользователи и права доступа, но в некоторых случаях возможно, что пользователь *A* в одной системе окажется пользователем *B* в другой. Это становится настоящей проблемой, когда эти системы совместно используют данные, так как данные, которые разрешено прочитать пользователю *A* одной системы, также будут доступны пользователю *B* другой системы.

Поэтому во многих организациях используется централизованная база данных пользователей, которая позволяет решить проблему пересечения списков пользователей в разных системах.

Домашние каталоги

Ещё один вопрос, возникающий перед системными администраторами – хранить ли домашние каталоги пользователей централизованно.

Основное преимущество централизованного хранения домашних каталогов на сетевом сервере состоит в том, что пользователь может обращаться к файлам в своём домашнем каталоге с любого компьютера в сети.

Но у этого подхода есть и недостаток – если работа сети будет нарушена, все пользователи потеряют доступ к своим файлам. В некоторых ситуациях (например, если в организации широко используются ноутбуки), иметь централизованные домашние каталоги может быть нежелательно. Но если это имеет смысл в вашей организации, развёртывание централизованных домашних каталогов может сильно упростить жизнь системного администратора.

Каким образом предотвращается нецелевое использование ресурсов?

Самое основное, что может сделать системный администратор для предотвращения нецелевого использования ресурсов пользователями – тщательно продумать организацию групп и осмотрительно распределять доступ к ресурсам. Таким образом, те, кто не должен иметь доступа к важным ресурсам, его не получат.

Но вне зависимости от того, как работает ваша организация, лучшая защита от злоупотребления ресурсами – постоянная бдительность системного администратора. Часто единственный способ избежать неприятного сюрприза, который может ожидать вас на рабочем месте в одно прекрасное утро – всегда быть начеку.

Приложение

Администрирование UNIX систем

Ниже представлены различные команды системной оболочки Unix подобных операционных систем. Имейте в виду, это просто сборник, не относящийся к какому-то конкретному дистрибутиву, то есть многие из этих команд, специфичны для какой-то определенной системы и могут отсутствовать в других дистрибутивах Linux. Большинство из приведенных ниже команд, являются достаточно повседневными, то есть часто используются в процессе администрирования Unix систем (различных дистрибутивов Linux)

Корректные: выключение, перезагрузка, выход из системы

init 0
Выключить систему
logout
Завершить текущую сессию
reboot
Перезагрузка
shutdown -h now
Еще один вариант корректного выключения
shutdown -h 22:15 &
Запланировать выключение системы на 22 часа 15 минут
shutdown -c
Отменить запланированное выключения системы
shutdown -r now
Еще один вариант перезагрузки
telinit 0
Тоже способ выключения системы

Получение различной информации о системе

arch
Вывести на экран архитектуру компьютера
uname -m
Так-же выводит архитектуру компьютера
cal 2010
Печатает календарь на 2010 год. Без аргументов выводит календарь на текущий месяц
cat /proc/cpuinfo
Вывести подробную информацию о процессоре
cat /proc/interrupts
Вывести информацию о прерываниях
cat /proc/meminfo
Выводит статистику использования памяти
cat /proc/swaps
Вывести информацию о swap файле(ах) (файл подкачки)

cat /proc/version

Вывести информацию о версии текущего ядра

cat /proc/net/dev

Вывести информацию и статистику по сетевым устройствам

cat /proc/mounts

Показать смонтированные файловые системы

clock -w

Записать текущую системную дату в BIOS

date

Вывести текущую системную дату и время

date 041217002007.00

Установить дату и время в значение МесяцДеньЧасМинутаГод.Секунда

dmidecode -q

Вывести в читабельном виде информацию по аппаратному оборудованию системы (SMBIOS / DMI)

hdparm -i /dev/hda

Вывести характеристики жесткого диска

hdparm -tT /dev/sda

Измерять скорость чтения данных с жесткого диска

lspci -tv

Вывести список устройств на шине PCI

lsusb -tv

Вывести список устройств на USB шине

uname -r

Вывести версию используемого ядра

Дисковое пространство

df -h

Вывод информации о свободном и занятом дисковом пространстве на смонтированных разделах, в формате, удобном для чтения

dpkg-query -W -f='${Installed-Size;10}t${Package}n' | sort -k1,1n

Выводит объем используемого дискового пространства, занятого файлами deb-пакета, с сортировкой по размеру (*ubuntu*, *debian* т.п.)

du -sh dir

Выводит объем дискового пространства, занимаемый директорией *dir*

du -sk * | sort -rn

Выводит листинг файлов и директорий с размером, сортируя его по размеру

ls -lSr | more

Выводит листинг файлов и директорий, сортируя по возрастанию размера и перенаправляет его в программу *more* для постраничного просмотра

rpm -q -a --qf '%10{SIZE}t%{NAME}n' | sort -k1,1n

Выводит объем дискового пространства, занимаемого файлами rpm-пакета, с сортировкой по размеру (*fedora*, *redhat* и т.п.)

Работа с файлами и папками

pwd

Вывести текущую директорию

cd /home

Перейти в директорию */home*

cd ..

Перейти в родительский каталог, то есть подняться на уровень выше

cd ../..

Подняться в дереве каталогов на два уровня

cd

Перейти в домашний каталог

cd ~user

Перейти в домашний каталог пользователя *user*

cd -

Перейти в предыдущую директорию, из которой был сделан переход в текущую директорию

cp -a test_1 test_2

Копировать директорию *test_1* в директорию *test_2*

cp file_1 file_2

Копировать файл file_1 в файл file_2

cp -a /tmp/test .

Копировать директорию *test* и все ее содержимое в текущую директорию

cp test/* .

Копировать все файлы содержащиеся в директории *test* в текущую директорию

iconv -l

Выводит список доступных для использования кодировок

iconv -c -f windows-1251 -t utf-8 inputFile > outputFile

Конвертировать файл из кодировки windows-1251 в кодировку utf-8

ln -s /path/to/filedir ./link

Создать в текущей директории символическую ссылку *link* на директорию или файл *filedir*

ln /path/to/filedir ./link

Создать в текущей директории жесткую ссылку *link* на директорию или файл *filedir*

ls

Вывести листинг содержимого текущей директории

ls -F

Листинг текущей директории с добавлением к именам символов, обозначающих тип объекта

ls -l

Листинг текущей директории с подробностями по каждому объекту (права доступа, владелец, группа, дата, размер)

ls -a

Кроме обычных объектов, вывести в листинге, скрытые файлы и директории, текущего каталога

ls /tmp | pr -T5 -W$COLUMNS

Листинг директории */tmp* в 5 колонок

ls *[0-9]*

Вывести в листинг файлов и директорий текущего каталога, содержащие в имени цифровые символы

lstree

Листинг иерархии директорий и файлов, начиная с корневого каталога *(/)*

mkdir dir

Создать в текущем каталоге, директорию с именем *dir*

mkdir dir_1 dir_2

Создать в текущем каталоге две директории, с именами *dir_1* и *dir_2*

mkdir -p /path/to/dir

Создать иерархию директорий. Кроме самой директории *dir* будут созданы все промежуточные директории

mv filedir new_filedir

Переместить (переименовать) файл или директорию из *filedir* в *new_filedir*

rm -rf /dir

Рекурсивно удалить директорию *dir* и всё её содержимое, без запроса подтверждения на удаление

rm -f file

Удалить файл *file*, без запроса подтверждения

rm -rf /dir_1 /dir_2

Удалить директории */dir_1* и */dir_2* вместе с содержимым, без запроса подтверждения

rmdir /dir

Удалить директорию */dir*

tree

Вывести иерархию директорий и файлов, начиная с корневой *(/)*

touch -t 0712250000 file1

Изменить дату и время создания файла, если указанный файл не существует, создать его с указанными датой и временем (YYMMDDhhmm)

Поиск нужного файла

find / -user vasya

Найти все директории и файлы, принадлежащие пользователю *vasya*. Поиск будет производиться, начиная с корневой директории *(/)*

find / -name filedir

Найти директории и файлы с именем *filedir*. Поиск будет производиться, начиная с корневой директории *(/)*

find /usr/bin -type f -atime +20

Найти все файлы в директории */usr/bin*, время последнего обращения к которым более 20 дней

find /usr/bin -type f -mtime -10

Найти все файлы в директории */usr/bin*, которые были созданы или модифицированы в течении последних 10 дней

find / -xdev -name *.rpm

Искать директории и файлы, имена которых оканчиваются на ".rpm", но игнорируя съёмные накопители, cdrom, usb и т.п.

find . -maxdepth 1 -name *.jpg -print -exec convert

Найти в текущей директории все файлы с расширением jpg и выполнить для каждого команду convert (Imagemagick)

find /path/to/dir -name *.c

Найти все директории и файлы, имена которых оканчиваются на ".c". Поиск будет производиться, начиная с директории */path/to/dir*

find / -name *.rpm -exec chmod 755 '{}' \;

Начиная с корня, найти все фалы и директории, имена которых оканчиваются на ".rpm", и для каждого изменить права доступа

whereis ls

Вывод полных путей к бинарным файлам, файлом исходных кодов и руководств, имеющих отношение к файлу *ls*

locate *.ps

Поиск всех файлов, содержащих в имени ".ps" по специальной базе данных, предварительно сформированной командой *updatedb*

which ls

Вывести полный путь до файла *ls*

Монтирование файловых систем

mount /dev/hda1 /mnt/disk

Монтировать устройство */dev/hda1* в папку с именем */mnt/disk*. точка монтирования, папка /mnt/disk, должна существовать

fuser -km /mnt/hda1

Размонтировать устройство *hda1* если оно заблокировано каким либо процессом

mount -o loop file.iso /mnt/cdrom

Смонтировать фаил или образ ISO в директорию */mnt/cdrom*

mount /dev/hdb /mnt/cdrecorder

Монтировать cdrw или dvdrom

mount -t vfat /dev/hda5 /mnt/hda5

Смонтировать раздел с файловой системой FAT32

mount /dev/cdrom /mnt/cdrom

Монтировать cdrom или dvdrom

mount /dev/fd0 /mnt/floppy

Монтировать floppy disk (дисковод мягких дисков)

mount /dev/hdc /mnt/cdrecorder

Монтировать cdrw или dvdrom

mount /dev/sda1 /mnt/usbdisk

Монтировать USB устройство

mount -t smbfs -o username=user,password=pass //WinClient/share /mnt/share

Монтировать сетевую файловую систему Windows (SMB/CIFS)

umount -n /mnt/hda2

Размонтировать без записи в файл */etc/mtab*, используется в ситуациях, когда файл находится в режиме только чтение или жесткий диск переполнен

umount /dev/hda2

Размонтировать диск *hda2*, предварительно неоходимо выйти из точки монтирования */mnt/hda2*

Пользователи и группы

useradd -c "User Vasya" -g admin -d /home/vasya -s /bin/bash vasya

Создать пользователя vasya, домашним каталогом будет создана директория */home/vasya*, в качестве системной оболочки будет назначен */bin/bash*, пользователь будет включен в группу admin, кроме того для учетной записи будет создан комментарий "User Vasya"

useradd vasya

Создать пользователя vasya

usermod -c "User FTP" -g system -d /ftp/ftpuser -s /bin/nologin ftpuser

Изменение пользователя

userdel -r vasya

Удалить пользователя с именем vasya и его домашнюю директорию

groupadd [group_name]

Создать группу с именем *group_name*

chage -E 2005-12-31 vasya

Установить дату окончания действия учётной записи пользователя vasya

groupdel [group_name]

Удалить группу *group_name*

groupmod -n test new_test

Переименовать группу *test* в *new_test*

newgrp - [group]

Изменяет основную группу текущего пользователя. При указании "-", ситуация будет идентичной той, в которой пользователь вышил из системы и снова вошёл. Если не указывать группу, основная группа будет назначена из файла */etc/passwd*

grpck

Проверка корректности системных файлов учётных записей. Проверяется файл/etc/group

pwck

Проверка корректности системных файлов учётных записей. Проверяются файлы */etc/passwd* и */etc/shadow*

passwd

Изменить пароль текущего пользователя

passwd vasya

Изменить пароль пользователя vasya (может выполнять только root)

Атрибуты файлов

chattr +a file

Разрешает только добавление данных в файл

chattr +d file

Игнорировать данный файл при создании резервной копии с помощью программы *dump*

chattr +c file

Разрешить ядру автоматически сжимать/разжимать содержимое файла

chattr +i file1

Защита файла от каких либо изменений или манипуляций: редактирование, перемещение, удаление, создание ссылок на него

chattr +S file1

Определяет, будет-ли при сохранении изменений, произведена синхронизация, как при использовании команды sync

chattr +u file1

При удалении файла с данным атрибутом, его будет сохранено, что-бы оставить пользователю возможность восстановить данные в случае необходимости

chattr +s file1

Позволяет безвозвратное удаление данного файла. При удалении файла с этим атрибутом, место на диске, занимаемое файлом, перезаписывается нулями, после чего файл уже не подлежит восстановлению

lsattr

Листинг файлов с атрибутами

Работа с правами доступа файлов и директорий

chown vasya /file

Назначить пользователя *vasya* владельцем файла *file*

chown -R vasya directory

Рекурсивно обойти директорию *directory* и назначить пользователя *vasya* владельцем всех вложенных файлов и директорий

chown vasya:group /file

Назначить владельца и группу для файла */file*

chmod ugo+rwx /directory

Установить полные права доступа *rwx (Read Write eXecute)* для всех *ugo (User Group Other)* на директорию */directory*. То-же самое можно сделать, используя числовой представление *chmod 777 directory*

chmod go-rwx /directory

Удалить все права на директорию */directory* для *группы* и *остальных*

chgrp new_group file

Изменить группу-владельца для *file* на *new_group*

chmod o+t /home/public

Установить так называемый *STIKY*-бит на директорию */home/public*. Удалить файл в такой директории может только владелец данного файла

chmod o-t /home/public

Удалить STIKY-бит с директории /home/public

chmod u+s /bin/binary_file

Установить SUID-бит на файл */bin/binary_file*. Это позволяет любому пользователю системы, запускать данный файл с правами *владельца файла*

chmod u-s /bin/binary_file

Удалить SUID-бит с файла */bin/binary_file*

chmod g+s /home/public

Установить SGID-бит на директории /home/public

chmod g-s /home/public

Удалить SGID-бит с директории /home/public

find / -perm -u+s

Поиск всех файлов с установленным SUID битом, начиная с корня файловой системы

ls -lh

Листинг текущего каталога с правами доступа

Архивация и сжатие файлов

gzip -9 file1

Поместить файл *file1* в архив *gzip* с максимальной степенью сжатия

rar a file1.rar file1 file2 dir1

Создать rar архив *file1.rar*, заархивировав файлы: *file1*, *file2* и директорию: *dir1*

rar a file1.rar test_file

Упаковать в rar архив *file1.rar* файл *test_file*

rar x file.rar

Разархивировать rar архив *file.rar*

bzip2 file1

Сжимает файл *file1*

bunzip2 file1.bz2

Разжимает файл *file1.bz2*

gzip file1

Сжимает файл *file1*

gunzip file1.gz

Разжимает файл *file1.gz*

tar -cvf archive.tar file1 file2 dir1

Создать tar архив *archive.tar*, упаковав в него файлы *file1*, *file2* и директорию *dir1*

tar -cvf archive.tar file

Упаковать в *tar*-архив *archive.tar*, файл *file*

tar -tf archive.tar

Вывести содержимое *tar* архива

tar -xvf archive.tar

Распаковать *tar* архив

tar -xvf archive.tar -C /tmp

Распаковать архив в */tmp*

tar -cvfz archive.tar.gz dir1

Создать *tar* архив и сжать его с помощью программы *gzip*

tar -xvfz archive.tar.gz

Разжать *tar* архив и распаковать его

tar -cvfj archive.tar.bz2 dir1

Создать архив и сжать его с помощью *bzip2* (ключ -j работает не во всех **nix* системах)

tar -xvfj archive.tar.bz2

Разжать архив и распаковать его (ключ -j работает не во всех **nix* системах)

zip file1.zip file1

Создать сжатый zip-архив

zip -r file1.zip file1 file2 dir1

Запаковать в архив несколько файлов и/или директорий

unzip file1.zip

Разжать и распаковать zip-архив

unrar x file1.rar

Распаковать rar-архив

Работа с RPM пакетами (Fedora, Red Hat и им подобные дистрибутивы)

rpm -e [package]

Удалить пакет *package*

rpm -qa | grep httpd

Вывести список установленных в системе пакетов и отобрать, содержащие в своем имени *httpd*

rpm -qa

Вывести список всех установленных в системе пакетов

rpm -qi [package]

Вывести информацию о пакете *package*

rpm -ivh [package.rpm]

Установить пакет с выводом сообщений и прогресс-бара

rpm -U [package.rpm]

Обновить пакет без изменений конфигурационных файлов, в случае отсутствия пакета, он будет установлен

rpm -ivh --nodeeps [package.rpm]

Установить пакет с выводом сообщений и прогресс-бара без контроля зависимостей

rpm -F [package.rpm]

Обновить пакет только если он установлен

rpm -q [package] --whatprovides

Список предоставляемой функциональности

rpm -q [package] --changelog

Вывести историю ревизий пакета

rpm -q [package] --scripts

Отобразит скрипты, запускаемые при установке/удалении пакета

rpm -qf /etc/httpd/conf/httpd.conf

Проверить какому пакету принадлежит указанный файл. Указывать следует полный путь и имя файла

rpm -qg "System Environment/Daemons"

Отобразить пакеты входящие в группу пакетов (fedora, redhat)

rpm -qc [package]

Вывести список конфигурационных файлов, входящих в пакет

rpm -ql [package]

Вывести список файлов, входящих в пакет

rpm -q [package] --whatrequires

Вывести список пакетов, необходимых для установки конкретного пакета по зависимостям

rpm -qp [package.rpm] -l

Отображает список файлов, входящих в пакет, но ещё не установленных в систему

rpm -Va

Проверить содержимое всех пакеты установленные в систему. Выполняйте с осторожностью!

rpm -ivh /usr/src/redhat/RPMS/`arch`/[package.rpm]

Установить пакет, собранный из исходных кодов

rpm -Vp [package.rpm]

Проверить пакет, который ещё не установлен в систему

rpm2cpio [package.rpm] | cpio --extract --make-directories *bin*

Извлечь из пакета файлы содержащие в своём имени *bin*

rpm --import /media/cdrom/RPM-GPG-KEY

Импортировать публичный ключ цифровой подписи

rpm --checksig [package.rpm]

Проверит подпись пакета

rpm -qa gpg-pubkey

Проверить целостность установленного содержимого пакета

rpm -V [package]

Проверить размер, полномочия, тип, владельца, группу, MD5-сумму и дату последнего изменения пакета

rpmbuild --rebuild [package.src.rpm]

Собрать пакет из исходных кодов

Средство управления пакетами – YUM (Fedora, RedHat и т.д.)

yum list

Вывести листинг пакетов, установленных в системе

yum clean headers

Удалить все заголовки файлов, которые система использует для разрешения зависимостей

yum clean [package]

Очистить rpm-кэш, удалив закачанные пакеты

yum search [package]

Найти пакет в репозитории

yum clean all

Очистить rpm-кэш, удалив закачанные пакеты и заголовки

yum -y install [package]

Скачать и установить пакет

yum update [package]

обновить пакет

yum -y update

Обновить все пакеты, установленные в систему

yum localinstall [package.rpm]

Попытаться установить пакет RPM и все зависимые от него пакеты, используя ваши репозитории

yum remove [package]

Удалить пакет

Средства управления DEB пакетами (Debian, Ubuntu и т.д.)

dpkg -l
Список пакетов, установленных в системе

dpkg -r [package]
Удалить пакет из системы

dpkg -i [package.deb]
Установить / обновить пакет

dpkg -l | grep httpd
Вывести список установленных в системе пакетов, отобрав, содержащие в своем названии *httpd*

dpkg -s [package]
Вывести информацию о конкретном пакете

dpkg --contents [package.deb]
Вывести список файлов, входящих в пакет, который ешё не установлен в систему

dpkg -L [package]
Вывести список файлов, входящих в пакет, установленный в систему

dpkg -S /bin/ping
В какой пакет входит указанный файл.

Система управления пакетами Pacman (Arch, Frugalware and alike)

pacman -S name
Установить пакет *name* со всеми зависимостями

pacman -R name
Удалить пакет и все его файлы

Средство управление пакетами – APT (Debian, Ubuntu и т.д.)

apt-cache search [package]
Вывести список пакетов, чье имя совпадает со строкой *package*

apt-get check
Проверить зависимости

apt-cdrom install [package]
Установить / обновить пакет с cdrom'a

apt-get install [package]
Установить / обновить пакет

apt-get upgrade
Обновить установленные в систему пакеты

apt-get remove [package]
Удалить установленный пакет из системы, сохранив файлы конфигурации

apt-get update
Обновить списки пакетов репозитария

apt-get clean
Удалить загруженные архивные файлы пакетов

Анализ файловой системы

badblocks -v /dev/hda1

Проверить раздел *hda1* на наличие *bad*-блоков

fsck /dev/hda1

Проверить и при необходимости попробовать восстановить целостность *linux*-файловой системы раздела *hda1*

fsck.ext2 /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы ext2 раздела *hda1*

fsck.msdos /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *fat* раздела *hda1*

fsck.vfat /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *fat* раздела *hda1*

fsck.ext3 /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *ext3* раздела *hda1*

dosfsck /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *fat*, раздела *hda1*

e2fsck /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *ext2* раздела *hda1*

e2fsck -j /dev/hda1

Проверить и при необходимости попробовать восстановить целостность файловой системы *ext3* раздела *hda1*, журнал файловой системы расположен там же

Просмотр содержимого файлов

cat file1

Вывести все содержимое файла начиная с первой строки

head -2 file1

Отобразить две первые строки файла

tac file1

Отобразить содержимое файла начиная с последней строки

tail -f /var/log/messages

В реальном времени выводить все, что добавляется в файл

tail -2 file1

Вывести две последние строки файла

more file1

Отобразить содержимое файла постранично

less file1

Аналогична команде *more* но позволяет перемещаться по содержимому вперед и назад

Манипуляции с текстом

cat -n file1

Вывести содержимое файла, нумеруя выводимые строки

cat example.txt | awk 'NR%2==1'

Вывести только не четные строки файла

echo a b c | awk '{print $1,$3}'

Вывести первую и третью колонки. Разделение, по-умолчанию, по пробельным символам или символу табуляции

echo a b c | awk '{print $1}'

Вывести первую колонку содержимого файла. Разделение, по-умолчанию, по пробельным символам или символу табуляции

comm -3 file1 file2

Сравнить содержимое двух файлов, удаляя строки встречающиеся в обоих файлах

comm -1 file1 file2

Сравнить содержимое двух файлов, не отображая строки принадлежащие файлу *file1*

comm -2 file1 file2

Сравнить содержимое двух файлов, не отображая строки принадлежащие файлу *file2*

grep [0-9] /var/log/messages

Отобрать и вывести строки содержащие цифровые символы из файла */var/log/messages*

grep ^Aug /var/log/messages

Отобрать и вывести строки, начинающиеся с сочетания символов *"Aug"*, из файла */var/log/messages*

grep Aug /var/log/messages

Отобрать и вывести строки, содержащие сочетание символов *"Aug"* из файла */var/log/messages*

grep Aug -R /var/log/*

Отобрать и вывести строки, содержащие сочетание символов *"Aug"*, из всех файлов, расположенных в директории */var/log* и ниже

paste -d '+' file1 file2

Объединить содержимое *file1* и *file2* в виде таблицы с разделителем "+"

paste file1 file2

Объединить содержимое *file1* и *file2* в виде таблицы: строка 1 из file1 = строка 1 колонка 1-n, строка 1 из file2 = строка 1 колонка n+1-m

sdiff file1 file2

Сравнить содержимое двух файлов

sed 's/string1/string2/g' example.txt

Заменить *string1* на *string2* в файле *example.txt* и вывести содержимое

sed '/ *#/d; /^$/d' example.txt

Удалить пустые строки и комментарии из файла *example.txt*

sed '/^$/d' example.txt

Удалить пустые строки и комментарии из файла *example.txt*

sed -e '1d' exampe.txt

Удалить первую строку из файла *example.txt*

sed -n '/string1/p'

Отобразить только строки содержащие *string1*

sed -e 's/string//g' example.txt

Удалить строку *string1* из текста файла *example.txt* не изменяя всего остального

sed -e 's/ *$//' example.txt

Удалить пустые символы в конце каждой строки файла *example.txt*

sed -n '5p;5q' example.txt

Вывести пятую строку

sed -n '2,5p' example.txt

Вывести строки со второй по пятую

sed -e 's/00*/0/g' example.txt

Заменить последовательность из любого количества нулей одним нулём

sort file1 file2

Вывести отсортированное содержимое двух файлов

sort file1 file2 | uniq

Вывести отсортированное содержимое двух файлов исключая повторные значения

sort file1 file2 | uniq -u

Вывести уникальные значения из отсортированного содержимого двух файлов

sort file1 file2 | uniq -d

Вывести только повторяющиеся значения из отсортированного содержимого двух файлов

echo 'word' | tr '[:lower:]' '[:upper:]'

Перевести символы нижнего регистра в верхний

Конвертирование содержимого текстовых файлов

dos2unix filedos.txt fileunix.txt

Конвертировать содержимое текстового файла из *MSDOS* кодировки в UNIX кодировку (разница в символах возврата коретки)

unix2dos fileunix.txt filedos.txt

Конвертировать содержимое текстового файла из *UNIX* кодировки в *MSDOS* кодировку (разница в символах возврата коретки)

recode ..HTML < page.txt > page.html

Конвертировать содержимое тестового файла *page.txt* в *html*-файл *page.html*

recode -l | more

Вывести список доступных форматов

Файловая система SWAP (файл подкачки)

mkswap /dev/hda3

Создание swap-пространство на разделе *hda3*

swapon /dev/hda3

Включить swap-пространство, расположенное на разделе *hda3*

swapon /dev/hda2 /dev/hdb3

Активировать swap-пространства, расположенные на разделах *hda2* и *hdb3*

Форматирование файловой системы

fdformat -n /dev/fd0

Форматирование флоппи-диска без проверки

mkfs /dev/hda1

Создать файловую систему *linux* на разделе *hda1*

mke2fs -j /dev/hda1

Создать журналируемую файловую систему *ext3* на разделе *hda1*

mke2fs /dev/hda1

Создание файловой системы *ext2* на разделе *hda1*

mkfs -t vfat 32 -F /dev/hda1

Создать файловую систему *FAT32* на разделе *hda1*

Резервное копирование (Backup)

find /var/log -name '*.log' | tar cv --files-from=- | bzip2 > log.tar.bz2

Поиск всех файлов, заканчивающихся на ".log" в директории */var/log*, и упаковка их в *bzip*-архив

find /home/user -name '*.txt' | xargs cp -av --target-directory=/home/backup/

--parents

Поиск в директории */home/user* файлов, имена которых оканчиваются на ".txt", и копирование их в другую директорию

rsync -rogpav --delete /home /tmp

Синхронизировать директории */tmp* и */home*

rsync -az -e ssh --delete ip_addr:/home/public /home/local

Синхронизировать локальную и удаленную директории через *ssh* туннель используя сжатие

rsync -rogpav -e ssh --delete /home ip_address:/tmp

Синхронизация через *SSH* туннель

rsync -az -e ssh --delete /home/local ip_addr:/home/public

Синхронизировать удалённую директорию с локальной используя *ssh* туннель со сжатием

dd bs=1M if=/dev/hda | gzip | ssh user@ip_addr 'dd of=hda.gz'

Создать "слепок" локального диска в файл на удалённом сервере используя *ssh* туннель

dd if=/dev/hda of=/dev/fd0 bs=512 count=1

Создание копии *MBR (Master Boot Record)* с */dev/hda* на флоппи-диск

dd if=/dev/sda of=/tmp/backup

Создание резервной копии содержимого жесткого диска в файл *backup*

dd if=/dev/fd0 of=/dev/hda bs=512 count=1

Восстановить *MBR* с флоппи-диска на */dev/hda*

dump -0aj -f /tmp/home0.bak /home

Создать полную резервную копию директории */home* в файл */tmp/home0.bak*

dump -1aj -f /tmp/home0.bak /home

Создать инкрементную резервную копию директории */home* в файл */tmp/home0.bak*

restore -if /tmp/home0.bak

Восстановить данные из резервной копии */tmp/home0.bak*

tar -Puf backup.tar /home/user

Создать инкрементную резервную копию директории */home/user* в файл *backup.tar* сохраняя права доступа

(cd /tmp/local/ && tar c .) | ssh -C user@ip_addr 'cd /home/share/ && tar x -p'

Упаковка в архив и копирование содержимого */tmp/local* в директорию */home/share/* удалённого сервера, используя *ssh* туннель

(tar c /home) | ssh -C user@ip_addr 'cd /home/backup-home && tar x -p'

Упаковка в архив и копирование содержимого */home* в директорию */home/backup-home* удалённого сервера, используя *ssh* туннель

tar cf - . | (cd /tmp/backup ; tar xf -)

Упаковка в архив и копирование одной директории в другую с сохранением прав доступа и ссылок

CDROM

cd-paranoia -B

Перенести аудио-треки с компакт-диска в wav-файлы.

cd-paranoia --

Перенести три аудио-трека с компакт-диска в wav-файлы.

cdrecord -v dev=/dev/cdrom cd.iso

Записать ISO-образ на компакт-диск.

gzip -dc cd_iso.gz | cdrecord dev=/dev/cdrom -

Записать сжатый ISO-образ на компакт-диск.

mkisofs /dev/cdrom > cd.iso

Создать ISO-образ компакт-диска.

```
# mkisofs -J -allow-leading-dots -R -V
```

Создать ISO-образ из содержимого директории.

```
# mkisofs /dev/cdrom | gzip > cd_iso.gz
```

Создать сжатый ISO-образ компакт-диска.

```
# mount -o loop cd.iso /mnt/iso
```

Смонтировать ISO-образ компакт-диска в файловую систему.

```
# cdrecord -v gracetime=2 dev=/dev/cdrom -eject blank=fast -force
```

Очистить перезаписываемый компакт-диск.

```
# cdrecord --scanbus
```

Сканировать системную шину для поиска идентификаторов SCSI каналов.

```
# dd if=/dev/hdc | md5sum
```

Вычислить контрольную сумму *MD5* для устройства, например, компакт-диска.

Сети (LAN / WiFi)

```
# dhclient eth0
```

Включить *DHCP* на сетевом интерфейсе *eth0*

```
# ethtool eth0
```

Вывод статистики по сетевому интерфейсу *eth0*

```
# hostname
```

Вывести имя компьютера

```
# host www.example.com
```

Преобразовать домен *www.example.org* в *ip*-адрес и наоборот

```
# ifconfig eth0
```

Вывести настройки сетевой карты *eth0*

```
# ifconfig eth0 promisc
```

Переключить интерфейс *eth0* в promiscuous-режим для сбора (сниффинг) сетевых пакетов

```
# ifup eth0
```

Включить сетевой интерфейс *eth0*

```
# ifdown eth0
```

Отключить сетевой интерфейс *eth0*

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
```

Назначить IP адрес и маску сетевому интерфейсу *eth0*

```
# ip link show
```

Вывести статус связи всех сетевых интерфейсов

```
# iwconfig eth1
```

Вывести конфигурацию беспроводного сетевого интерфейса *eth1*

```
# iwlist scan
```

Сканирование и поиск беспроводных сетей и точек доступа

```
# mii-tool eth0
```

Вывести состояние связи сетевого интерфейса *eth0*

```
# nslookup www.example.com
```

Ресольвить (преобразовать/разрешить) доменное имя *www.example.org* в ip-адрес и наоборот

```
# route -n
```

Песать локальной таблицы маршрутизации

```
# route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.168.1.1
```

Добавить статический маршрут в сеть 192.168.0.0/16 через шлюз с ip-адресом 192.168.1.1

route add -net 0/0 gw IP_Gateway

Назначить ip-адрес шлюза по умолчанию *(default gateway)*

route del 0/0 gw IP_gateway

Удалить ip-адрес шлюза по умолчанию *(default gateway)*

netstat -tup

Выводит листинг всех установленных соединений по протоколам *TCP* и *UDP* без разрешения имён в ip-адреса а так-же *PID*'ы и имена процессов, обслуживающих данные соединения

netstat -tupl

Вывод списка соединений по протоколам *TCP* и *UDP* без разрешения имён в ip-адреса а так-же и *PID*'ы и имена процессов, ожидающих соединений на сетевых портах

netstat -rn

Вывести таблицу маршрутизации, аналог команды *route -n*

echo "1" > /proc/sys/net/ipv4/ip_forward

Разрешить форвардинг (пересылку) пакетов

tcpdump tcp port 80

Отлавливать и выводить весь трафик на TCP-порт 80 (обычно - HTTP)

whois www.example.com

Вывести информацию о доменном имени из базы данных *whois*

Microsoft Windows networks (samba)

mount -t smbfs -o username=user,password=pass //WinClient/share /mnt/share

Монтировать smb-ресурс, расшаренный на windows-машине, в папку локальной файловой системы

nbtscan ip_addr

Преобразовать имя *netbios*. Программа nbtscan не во всех системах присутствует по-умолчанию, nmblookup включен в пакет samba

nmblookup -A ip_addr

Преобразовать имя *netbios*. Программа nbtscan не во всех системах присутствует по-умолчанию, nmblookup включен в пакет samba

smbclient -L ip_addr/hostname

Вывести список ресурсов, выделенных в общий доступ на windows-машине

smbget -Rr smb://ip_addr/share

Аналог программы wget для SMB протокола

Фаервол IPTABLES, штатный, для большинства дистрибутивов Linux

iptables -t filter -L

Вывести список всец цепочек правил

iptables -t nat -L

Вывести все цепочки из *NAT* таблицы

iptables -t nat -F

Очистить все цепочки правил в таблице *NAT*

iptables -t filter -X

Очистить все пользовательские цепочки правил в таблице *filter*

iptables -t filter -F

Очистить все цепочки правил в таблице *filter*

iptables -t filter -A INPUT -p tcp --dport telnet -j ACCEPT

Разрешить входящие соединения с *telnet*

iptables -t filter -A OUTPUT -p tcp --dport http -j DROP

Запретить исходящие *HTTP* соединения

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

включить NAT (Network Address Translate) исходящих пакетов на интерфейс eth0. Допустимо при использовании с динамически выделяемыми ip-адресами.

iptables -t nat -A PREROUTING -d 192.168.0.1 -p tcp -m tcp --dport 22 -j DNAT --to-destination 10.0.0.2:22

Пересылка пакетов, адресованных одному хосту, на другой хост

iptables -t filter -A INPUT -j LOG --log-prefix

Включить логгирование пакетов, проходящих через цепочку *INPUT*, и добавлением к сообщению префикса *"DROP INPUT"*

iptables -t filter -A FORWARD -p tcp --dport pop3 -j ACCEPT

Разрешить форвардинг *POP3* соединений

Мониторинг и отладка системы

free -m

Вывод статистики по оперативной памяти

kill -9 proc_id

Убить процесс с PID *proc_id*, без соблюдения целостности данных, то есть насмерть

kill -1 proc_id

Перечитать файл конфигурации процессом с PID *proc_id*

last reboot

Вывод истории ребутов системы

lsof /home/user1

Вывести список открытых файлов из директории */home/user1*

lsof -p proc_id

Вывести список файлов, открытых процессом с PID *proc_id*

lsmod

Список загруженных модулей ядра

ps -e -o pid,args --forest

Вывести список PID'ов и процессов в виде дерева

ps -eafw

Отобразить работающие в системе процессы, используемые ими ресурсы и другую полезную информацию (единожды)

pstree

Вывести дерево процессов

smartctl -i /dev/hda

Проверить доступность *SMART* на жёстком диске */dev/hda*

smartctl -A /dev/hda

Проверка состояния жёсткого диска */dev/hda* через *SMART*

strace -c ls >/dev/null

Вывести список системных вызовов, созданных и полученных процессом ls

strace -f -e open ls >/dev/null

Вывести список вызовов системных бибилотек

tail /var/log/messages

Вывести десять последних записей из системного журнала

tail /var/log/dmesg

Вывести десять последних записей из журнала загрузки ядра

top

Вывести список работающих в системе процессов с различной полезной информацией в режиме реального времени с автоматическим обновлением данных

watch -n1 'cat /proc/interrupts'

Другие полезные команды

alias hh='history'

Создать псевдоним *hh* для команды *history*

apropos ...keyword

Вывод комманд, так или иначе относящихся к ключевым словам. Полезно, когда вы знаете что делает программа, но не помните команду

chsh

Изменить системную оболочку пользователя

gpg -c file1

Шифрует файл file1 с помощью GNU Privacy Guard

gpg file1.gpg

Дешифрует файл file1 с помощью GNU Privacy Guard

ldd /usr/bin/ssh

Список библиотек, используемых программой *ssh*

man ping

Ввывод страниц руководства по работе с программой, в данном случае, *ping*

mkbootdisk --device /dev/fd0 `uname -r`

Создаёт загрузочный флоппи-диск

wget -r www.example.com

Рекурсивно загружает содержимое сайта *www.example.com*

wget -c www.example.com/file.iso

Загрузить файл *www.example.com/file.iso* с возможностью остановки и докачки

echo 'wget -c www.example.com/files.iso' | at 09:00

Включить закачку в определенное время

whatis ...keyword

Вывести описание действий указанной программы

who -a

Вывести список залогиненных пользователей, время последней загрузки системы и прочую полезную информацию

Учебное издание

Администрирование вычислительных систем и сетей

Учебное пособие

Составитель
ШЕВЕЛЁВ Геннадий Ефимович
Научный редактор
доктор технических наук, *профессор* Лисин В.А.

В авторской редакции

Компьютерная верстка Г.Е. Шевелев

Национальный исследовательский Томский
политехнический университет

Система менеджмента качества

Издательства Томского политехнического университета
сертифицирована в соответствии с требованиями ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ 634050, г. Томск, пр. Ленина, 30.
Тел./факс: 8(3822)56-35-35, www.tpu.ru