

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра информационных систем
и программной инженерии

Х.М. Салех

ИНФОКОММУНИКАЦИОННЫЕ СИ- СТЕМЫ И СЕТИ

Методические указания к лабораторным работам

Владимир 2013

Содержание

Лабораторная работа №1	3
Лабораторная работа №2	37
Лабораторная работа №3	51
Лабораторная работа №4	70
Лабораторная работа №5	81
Лабораторная работа №6	90

ЛАБОРАТОРНАЯ РАБОТА №1 БАЗОВАЯ НАСТРОЙКА СЕТЕВЫХ СРЕДСТВ ОС

1. Цель работы

Получить навыки базовой настройки сетевых средств ОС Windows.

2. Общие сведения

2.1 Работа с командной строкой Windows

Консоль командной строки присутствует во всех версиях операционных систем Windows. Ранние версии ОС поддерживали режим MS-DOS напрямую, что позволяло выполнять простые команды прямо из консоли. Представители же семейства NT, такие как Windows 2000 или Windows Server 2003, работают уже совсем по другим принципам, однако MS-DOS в них тоже поддерживается, но через виртуальную машину (NT Virtual DOS Machine, NTVDM), что позволяет контролировать и администрировать системные ресурсы прямо из консоли командного режима. В качестве интерпретатора командного режима выступает программа cmd.exe, запуск которой осуществляется через меню «Пуск -> Выполнить». Кроме того, для запуска консоли можно воспользоваться элементом меню «Пуск -> Программы -> Стандартные -> Командная строка».

Запустив консоль командного режима, пользователь может управлять ресурсами как локальной системы, так и ресурсами удаленной машины. Существуют команды, выполняющие мониторинг системы и выявляющие критические места в настройках сервера. Отличием работы из командной строки является полное отсутствие больших и громоздких графических утилит. Программы командной строки позволяют более тонкую настройку в виде параметров-ключей, указанных справа от самой команды.

С помощью специальных файлов-скриптов (наборов команд, выполняющихся последовательно или в запрограммированном порядке) администратор может свести к минимуму выполнение рутинных ежедневных операций. Существующие современные утилиты могут запускать такие скрипты с заданной периодичностью без присутствия администратора системы.

Сам администратор может выполнять как одиночные команды, так и список команд, используя специальные управляющие символы (&, |). Например:

Команда 1 & Команда 2 — сначала будет выполнена Команда 1 и только затем Команда 2;

Команда 1 && Команда 2 — только после успешного выполнения

Команды 1 будет запущена Команда 2.

Существует возможность перенаправить выводимый программой поток напрямую в текстовый файл для дальнейшей обработки. Для этого необходимо использовать управляющий символ «>» и имя текстового файла. Пример вывода содержания текущего каталога в текстовый файл Report.txt при помощи команды dir приведен ниже:

```
dir> Report.txt
```

В современных операционных системах существует множество команд и утилит. Запомнить такое количество различных программ, а тем более их параметров очень сложно, поэтому одним из самых важных параметров для каждой программы является сочетание символов /?. Выполнив команду с таким параметром, пользователь получит исчерпывающее сообщение о применении утилиты и синтаксисе ее параметров.

Для того чтобы закрыть консоль командной строки, необходимо выполнить команду exit.

2.2 Утилиты командной строки

Большинство рассматриваемых сетевых утилит для полноценной работы требуют наличия административных привилегий. Для операционных систем семейства Windows 2000/XP достаточно того, чтобы пользователь работал под учетной записью члена группы администраторов. Интерпретатор командной строки **cmd.exe** можно запустить с использованием меню **Пуск - Выполнить - cmd.exe**. В среде операционных систем Windows Vista/Windows 7 интерпретатор **cmd.exe** должен быть запущен для выполнения с использованием пункта контекстного меню "Запустить от имени администратора". Командные файлы, в которых используются сетевые утилиты, также должны выполняться в контексте учетной записи с привилегиями администратора.

В списке представлены сетевые утилиты командной строки для получения информации о сетевых настройках, выполнения операций по конфигурированию и диагностике сети.

В описании команд используется

< текст > - текст в угловых скобках. Обязательный параметр.

[текст] - текст в квадратных скобках. Необязательный параметр.

(текст) - текст в круглых скобках. Необходимо выбрать один из параметров.

Вертикальная черта | - разделитель для взаимоисключающих параметров.

Нужно выбрать один из них.

Многоточие ... - возможно повторение параметров.

Основные утилиты:

1. Hostname.

Одна из основных утилит TCP/IP. Она выводит имя системы, на которой запущена команда:

C:\>hostname

2. Ping.

Команда Ping лежит в основе диагностики сетей TCP/IP. Если до системы не удастся «достучаться» с помощью этой команды, вероятнее всего, с такой системой связаться не удастся.

Синтаксис: ping [-t] [-a] [-n число] [-l размер] [-f] [-i TTL] [-v TOS]
[-r число] [-s число] [[-j списокУзлов] | [-k списокУзлов]]
[-w таймаут] конечноеИмя

Параметры:

-t Отправка пакетов на указанный узел до команды прерывания.

Для вывода статистики и продолжения нажмите <Ctrl>+<Break>, для прекращения - <Ctrl>+<C>.

-a Определение адресов по именам узлов.

-n число Число отправляемых запросов.

-i TTL Задание срока жизни пакета (поле "Time To Live").

-r число Запись маршрута для указанного числа переходов.

-j списокУзлов Свободный выбор маршрута по списку узлов.

-k списокУзлов Жесткий выбор маршрута по списку узлов.

-w таймаут Таймаут каждого ответа в миллисекундах.

Примеры:

Чтобы опросить станцию с IP-адресом 192.168.100.1, следует набрать:
C:\>ping 192.168.100.1

3. Tracert.

Эта команда используется для верификации пути через маршрутизатор между данной станцией и удаленной. Tracert фиксирует число переходов или «прыжков» (hop), которые потребовалось совершить на пути к станции назначения.

Синтаксис: tracert [-d] [-h максЧисло] [-j списокУзлов] [-w интервал] имя

Параметры:

-d Без разрешения в имена узлов.
-h максЧисло Максимальное число прыжков при поиске узла.
-j списокУзлов Свободный выбор маршрута по списку узлов.
-w интервал Интервал ожидания каждого ответа в миллисекундах.

Примеры:

Например, чтобы вывести трассу маршрута к <http://www.vtsnet.ru>, нужно набрать:
C:\>tracert www.vtsnet.ru

Трассировка маршрута к www.vtsnet.ru [81.27.51.133] с максимальным числом прыжков 30:

1	<1 ms	<1 ms	<1 ms	192.168.128.100
2	14 ms	15 ms	13 ms	192.168.149.217
3	17 ms	18 ms	18 ms	192.168.194.253
4	28 ms	22 ms	20 ms	www3.vtsnet.ru [81.27.51.133]

Трассировка завершена.

4. Pathping.

Одна из самых полезных новых команд диагностики TCP/IP. Она объединяет функциональность Ping и Tracert. Команда Pathping опрашивает каждый маршрутизатор на пути между источником и приемником сигнала, после чего фиксирует задержки при каждой ретрансляции сигнала и потери пакетов.

Синтаксис: pathping [-g Список] [-h Число_прыжков] [-i Адрес] [-n]
[-p Пауза] [-q Число_запросов] [-w Таймаут] [-P] [-R] [-T]
[-4] [-6] узел

Параметры:

-h Число_прыжков Максимальное число прыжков при поиске узла.
-p Пауза Пауза между отправками (мсек).
-q Число_запросов Число запросов при каждом прыжке.
-w Таймаут Время ожидания каждого ответа (мсек).
-4 Принудительно использовать IPv4.
-6 Принудительно использовать IPv6.

Примеры:

C:\Documents and Settings\avk.SMART>pathping www.vtsnet.ru

Трассировка завершена.

6

Для ключей /Release и /Renew, если не указано имя адаптера, то будет освобожден или обновлен IP-адрес, выданный для всех адаптеров, для которых существуют привязки с TCP/IP.

Примеры:

> ipconfig - Отображает краткую информацию.
> ipconfig /all - Отображает полную информацию.
> ipconfig /renew - Обновляет сведения для всех адаптеров.
> ipconfig /renew EL* - Обновляет сведения для адаптеров, начинающихся с EL....
> ipconfig /release *ELINK?21* - Освобождает IP-адреса для всех адаптеров, имена которых удовлетворяют запросу: ELINK-21 или myELELINKi21adapter и т.п.

6. Netstat.

Утилита netstat присутствует во всех версиях Windows, однако, существуют некоторые отличия используемых параметров командной строки и результатов ее выполнения, в зависимости от операционной системы. Используется для отображения TCP и UDP -соединений, слушаемых портов, таблицы маршрутизации, статистических данных для различных протоколов.

Команда Netstat показывает текущий статус и статистику подключений по TCP/IP или UDP. При этом выводятся данные как о локальных, так и об удаленных именах и портах активных сетевых соединений. Ключ ? показывает все доступные ключи при работе с Netstat.

Синтаксис: netstat [-a] [-b] [-e] [-n] [-o] [-p протокол] [-r] [-s] [-v] [интервал]

Параметры:

-a Отображение всех подключений и ожидающих портов.
-b Отображение исполняемого файла, участвующего в создании каждого подключения, или ожидающего порта. Иногда известные исполняемые файлы содержат множественные независимые компоненты. Тогда отображается последовательность компонентов, участвующих в создании подключения, либо ожидающий порт. В этом случае имя исполняемого файла находится снизу в скобках [], сверху - компонент, который им вызывается, и так до тех пор, пока не достигается TCP/IP. Заметьте, что такой подход может занять много времени и требует достаточных разрешений.
-e Отображение статистики Ethernet. Он может применяться вместе с параметром -s.
-n Отображение адресов и номеров портов в числовом формате.
-o - отображение соединений, включая идентификатор процесса (PID) для каждого соединения.
-p протокол Отображение соединений для заданного протокола. Протокол может принимать значения tcp, udp, tcpv6, udpv6 . При использовании совместно с параметром -s в качестве протокола можно задавать tcp, udp, icmp, ip, tcpv6, udpv6, icmpv6, ipv6.
-r Отображение содержимого таблицы маршрутов.
-s Отображение статистических данных по протоколам. По умолчанию данные отображаются для IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP и UDPv6. Параметр -p позволяет указать подмножество выводимых данных.
-v Отображать подробную информацию. При использовании с параметром -b, отображает последовательность компонентов, участвующих в создании подключения, или ожидающий порт для всех исполняемых файлов.

интервал Интервал обновления отображаемой информации в секундах. Повторный вывод статистических данных через указанный промежуток времени в секундах. Для прекращения вывода данных нажмите клавиши CTRL+C. Если параметр не задан, сведения о текущей конфигурации выводятся один раз.

/? - отобразить справку по использованию netstat

Команды:

При использовании утилиты **netstat.exe** удобно пользоваться командами постраничного вывода (more), перенаправления стандартного вывода в файл (>) и поиска текста в результатах (find).

netstat -a | more - отобразить все соединения в постраничном режиме вывода на экран.

netstat -a > C:\netstatall.txt - отобразить все соединения с записью результатов в файл C:\netstatall.txt.

netstat -a | find /I "LISTENING" - отобразить все соединения со статусом LISTENING. Ключ **/I** в команде **find** указывает, что при поиске текста не нужно учитывать регистр символов.

netstat -a | find /I "listening" > C:\listening.txt - отобразить все соединения со статусом LISTENING с записью результатов в файл C:\listening.txt.

Пример отображаемой информации:

Активные подключения

Имя	Локальный адрес	Внешний адрес	Состояние
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
[httpd.exe]			
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
Не удастся получить сведения о владельце			
TCP	0.0.0.0:5800	0.0.0.0:0	LISTENING
[WinVNC.exe]			
TCP	127.0.0.1:50197	127.0.0.1:50198	ESTABLISHED
[firefox.exe]			
UDP	192.168.0.107:1900	*:*	
SSDPSRV [svchost.exe]			
...			

Имя - название протокола.

Локальный адрес - локальный IP-адрес участвующий в соединении или связанный со службой, ожидающей входящие соединения (слушающей порт). Если в качестве адреса отображается 0.0.0.0 , то это означает - "любой адрес", т.е в соединении могут использоваться все IP-адреса существующие на данном компьютере. Адрес 127.0.0.1 - это петлевой интерфейс, используемый в качестве средства IP протокола для взаимодействия между процессами без реальной передачи данных.

Внешний адрес Внешний IP-адрес, участвующий в создании соединения.

Состояние - состояние соединения. Состояние **Listening** говорит о том, что строка состояния отображает информацию о сетевой службе, которая ожидает входящие соединения по соответствующему протоколу на адрес и порт, отображаемые в колонке "Локальный адрес ". Состояние **ESTABLISHED** указывает на активное соединение. В колонке "Состояние" для соединений по протоколу TCP может отображаться текущий этап TCP-сессии определяемый по обработке значений флагов в заголовке TCP - пакета (Syn, Ask, Fin ...). Возможные состояния:

CLOSE_WAIT - ожидание закрытия соединения.
CLOSED - соединение закрыто.
ESTABLISHED - соединение установлено.
LISTENING - ожидается соединение (слушается порт)
TIME_WAIT - превышение времени ответа.

Имя программного модуля, связанного с данным соединением отображается, если задан параметр **-b** в командной строке при запуске netstat.exe.

Примеры:

- Чтобы вывести все активные подключения, отсортированные по возрастанию номера порта, необходимо набрать:
C:\>netstat -n
- Получить список слушаемых портов и связанных с ними программ:

netstat -a -b

netstat -ab - параметры командной строки можно объединять. Параметр -ab эквивалентен -a -b

netstat -a -n -b - отобразить список всех соединений с числовыми номерами портов

netstat -anb - аналогично предыдущей команде.

netstat -anbv - при использовании параметра -v отображается последовательность компонентов, участвующих в создании соединения или слушаемого порта.

- Получить статистические данные:

netstat -e - получить статистические данные для Ethernet. Отображаются суммарные значения принятых и полученных байт для всех сетевых адаптеров.

netstat -e -v - кроме статистических данных, отображается информация о каждом сетевом адаптере.

netstat -e -s - дополнительно к статистике Ethernet, отображается статистика для протоколов IP, ICMP, TCP, UDP

netstat -s -p tcp udp - получить статистику только по протоколам TCP и UDP

7. Route.

Эта команда нужна для редактирования или просмотра таблицы маршрутов IP из командной строки. Windows 2000 использует таблицу маршрутов в том случае, когда нужно отыскать путь к удаленному компьютеру по TCP/IP. Ключ ? выводит все доступные ключи при работе с Route.

Синтаксис: route [-f] [-p] [команда [узел]
[MASK маска] [шлюз] [METRIC метрика] [IF-интерфейс]

Параметры:

-f Очистка таблиц маршрутов от записей для всех шлюзов. При указании одной из команд, таблицы очищаются до выполнения команды.

-p При использовании с командой ADD задает сохранение маршрута при перезагрузке системы. По умолчанию маршруты не сохраняются при перезагрузке. Игнорируется

для остальных команд, изменяющих соответствующие постоянные маршруты. Этот параметр не поддерживается в Windows 95.

команда	Одна из четырех команд
PRINT	Печать маршрута
ADD	Добавление маршрута
DELETE	Удаление маршрута
CHANGE	Изменение существующего маршрута
узел	Адресуемый узел.
METRIC	Определение параметра метрика/цена для адресуемого узла.

Примеры:

Для просмотра таблицы маршрутов системы используется Route Print:
C:\>route print

8. Arp.

Команда Arp используется для просмотра, добавления или удаления записей в таблицах трансляции адресов IP в физические адреса. Эти записи используются при работе протокола Address Resolution Protocol (ARP).

Синтаксис:

Отображение и изменение таблиц преобразования IP-адресов в физические, используемые протоколом разрешения адресов (ARP).

```
ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -a [inet_addr] [-N if_addr]
```

Параметры:

-a Отображает текущие ARP-записи, опрашивая текущие данные протокола. Если задан inet_addr, то будут отображены IP и физический адреса только для заданного компьютера. Если более одного сетевого интерфейса используют ARP, то будут отображаться записи для каждой таблицы.

-g То же, что и ключ -a.

inet_addr Определяет IP-адрес.

-N if_addr Отображает ARP-записи для заданного в if_addr сетевого интерфейса.

-d Удаляет узел, задаваемый inet_addr. inet_addr может содержать символ шаблона * для удаления всех узлов.

-s Добавляет узел и связывает internet адрес inet_addr с физическим адресом eth_addr. Физический адрес задается 6 байтами (в шестнадцатеричном виде), разделенных дефисом. Эта связь является постоянной.

eth_addr Определяет физический адрес.

if_addr Если параметр задан, - он определяет интернет адрес интерфейса, чья таблица преобразования адресов должна измениться. Если не задан, - будет использован первый доступный интерфейс.

Примеры:

arp -s 157.55.85.212 00-aa-00-62-c6-09 ... Добавляет статическую запись.

arp -a ... Выводит ARP-таблицу.

Чтобы просмотреть содержимое занесенных в кэш адресов IP и MAC-адресов конкретной системы, нужно набрать:

C:\>arp —a

9. Getmac.

Утилита командной строки GETMAC присутствует в версиях Windows XP и старше. Используется для получения аппаратных адресов сетевых адаптеров (MAC-адресов) как на локальном, так и на удаленном компьютере.

Синтаксис: getmac [/S <система> [/U <пользователь> [/P <пароль>]]] [/FO <формат>] [/NH] [/V]

Параметры:

/S <система> - имя или IP-адрес удаленного компьютера.

/U [<домен>]<пользователь> Имя пользователя. Если не задано, то используется текущая учетная запись.

/P [<пароль>] - Пароль. Если задан параметр /U и не задан пароль, то он будет запрошен.

/FO <формат> - Формат, в котором следует отображать результаты запроса. Допустимые форматы: "TABLE" (таблица), "LIST" (список), "CSV" (разделяемые запятыми поля). Если параметр не задан, то используется вывод в виде таблицы (TABLE).

/NH - Указывает, что строка заголовков столбцов не должна отображаться в результирующем файле. форматов TABLE и CSV.

/V - Отображение подробной информации. В отображаемой информации присутствует имя сетевого подключения и название сетевого адаптера.

/? - Вывод справки по использованию команды.

Примеры:

GETMAC /? - отобразить краткую справку об использовании GETMAC.

GETMAC /FO csv - выдать информации о MAC-адресах всех существующих на локальном компьютере сетевых адаптеров в формате CSV (полей с разделителями в виде запятой).

GETMAC /S COMPUTER /NH /V - получить MAC адреса сетевых адаптеров для удаленного компьютера COMPUTER, не отображать заголовки столбцов в таблице и использовать отображение подробной информации. Для подключения к удаленному компьютеру используется текущая учетная запись пользователя.

GETMAC /S 192.168.1.1 /NH /V - то же самое, но вместо имени компьютера задан его IP-адрес.

GETMAC /S COMPUTER /U user /P password - получить MAC - адрес адаптеров удаленного компьютера COMPUTER. Для подключения к нему используется имя пользователя "user" и пароль "password"

GETMAC /S COMPUTER /U mydomain\user - для подключения к удаленному компьютеру используется учетная запись пользователя "user" в домене "mydomain". Пароль пользователя вводится по запросу.

GETMAC /S COMPUTER /U mydomain\user /P password - то же самое, что и в предыдущем случае, но пароль задан в командной строке.

10. Nbtstat.

Команда NBTSTAT позволяет получить статистику протокола NetBIOS over TCP/IP (NetBT), таблицу имен локальных и удаленных компьютеров и содержимое кэш NetBIOS имен. Применение NBTSTAT позволяет принудительно обновить кэш NetBIOS-имен компьютеров и имена, зарегистрированные с помощью серверов Windows Internet Name Service

(WINS).

Синтаксис: nbtstat[-aRemoteName] [-AIPAddress] [-c] [-n] [-r] [-R] [-RR] [-s] [-S] [Interval]

Параметры:

-a RemoteName - отображает таблицу имен удаленного компьютера. NetBIOS-имена соответствуют перечню NetBIOS-приложений, выполняющихся на удаленном компьютере.

-A IPAddress - то же самое, что и в предыдущем случае, но вместо имени удаленного компьютера используется его IP-адрес

-c - отображает кэш имен NetBIOS и соответствующих им IP-адресов.

-n - отображает таблицу NetBIOS-имен на локальном компьютере. Состояние "Зарегистрирован" означает, что имя зарегистрировано с использованием широковещательного запроса или с помощью сервера WINS.

-r - отображает статистику разрешения NetBIOS-имен. На компьютерах под управлением Windows XP и старше, выдается отдельная статистика о разрешении имен с помощью широковещательной рассылки и с помощью сервера имен WINS.

-R - очистка кэш NetBIOS-имен и загрузка данных из секции #PRE файла LMHOSTS.

-RR - очистка кэш NetBIOS - имен на локальном компьютере и их повторная регистрация с использованием сервера WINS.

-s - отображает статистику NetBIOS - сессий между клиентом и сервером и NetBIOS-имена удаленных узлов.

-S - отображает статистику сессий между клиентом и сервером и IP-адреса удаленных узлов.

Interval - интервал обновления отображаемых данных в секундах. Для прекращения автоматического обновления используется комбинация клавиш CTRL+C

/? - отобразить справку по использованию NBTSTAT.

Примеры:

nbtstat -n - вывести список зарегистрированных NetBIOS-имен на локальном компьютере.

nbtstat -a SERVER - вывести список зарегистрированных NetBIOS-имен на компьютере SERVER.

nbtstat -A 192.168.1.1 - вывести список зарегистрированных NetBIOS-имен на удаленном компьютере с IP-адресом 192.168.1.1 .

nbtstat -RR - выполнить очистку и перерегистрацию NetBIOS-имен на локальном компьютере.

11. Netsh.exe

Утилита сетевой оболочки NETSH (NETwork SHell) - наиболее полное и функциональное стандартное средство управления сетью с использованием командной строки в среде Windows XP и старше. Набор внутренних команд сетевой оболочки пополняется с появлением новых версий операционной системы, что необходимо учитывать при работе в локальной сети с различными ОС. Так, например, команда уровня wlan (netsh wlan - управление беспроводной сетью) может использоваться на компьютерах под управлением Windows Vista и старше и отсутствует в Widows XP. Синтаксис используемых команд и параметров также может различаться в разных операционных системах семейства Windows.

При запуске NETSH.EXE без параметров на экран выводится приглашение к вводу внутренних команд оболочки. Набор команд представляет собой многоуровневую структуру, позволяющую выполнять необходимые действия в выбранном контексте. При вводе знака вопроса ? можно получить краткую справку по доступному перечню команд на данном уровне. Ввод команды данного уровня со знаком вопроса вызовет отображение справки по ее исполь-

зованию. Аналогичную справку можно получить, введя определенную команду и, после перехода на уровень ее выполнения, ввести знак вопроса. При необходимости, можно выполнить нужное действие без использования интерактивного режима, указав в качестве параметров командной строки последовательный набор внутренних команд NETSH и необходимых параметров.

Команды NETSH можно выполнить и на удаленном компьютере с использованием подключения по локальной сети. Netsh также предоставляет возможность выполнения сценариев, представляющих собой группу команд в текстовом файле, выполняемых в режиме очереди на определенном компьютере. В целом, возможности NETSH настолько обширны, что трудно найти сетевую задачу, которую невозможно было бы решить с использованием данной утилиты.

Синтаксис: NETSH.EXE [-a AliasFile] [-c Context] [-r RemoteMachine] [-u [DomainName\]UserName] [-p Password | *] [Command | -f ScriptFile]

Параметры:

-a AliasFile - не завершать работу а перейти к приглашению ввода команд после выполнения AliasFile. AliasFile - имя текстового файла, в котором содержатся одна или несколько команд netsh.

-c Context - изменить контекст (уровень) команд netsh.

-r RemoteMachine - выполнять команды netsh на удаленном компьютере. В качестве RemoteMachine может использоваться имя или IP-адрес.

[-u DomainName\]UserName - имя пользователя для подключения к удаленному компьютеру. Если не задано, то используется текущее имя пользователя.

-p Password пароль для подключения к удаленному компьютеру.

Command - команда оболочки netsh, которую необходимо выполнить.

-f ScriptFile - аналогично ключу -a, но после выполнения команд файла сценария Scriptfile, работа netsh завершается.

Команды утилиты:

? - Отображение списка команд.

add - Добавление элемента конфигурации в список элементов.

advfirewall - Изменения в контексте 'netsh advfirewall'.

branchcache - Изменения в контексте 'netsh branchcache'.

bridge - Изменения в контексте 'netsh bridge'.

delete - Удаление элемента конфигурации из списка элементов.

dhcpcclient - Изменения в контексте 'netsh dhcpcclient'.

dnsclient - Изменения в контексте 'netsh dnsclient'.

dump - Отображение сценария конфигурации.

exec - Запуск файла сценария.

firewall - Изменения в контексте 'netsh firewall'.

help - Отображение списка команд.

http - Изменения в контексте 'netsh http'.

interface - Изменения в контексте 'netsh interface'.

ipsec - Изменения в контексте 'netsh ipsec'.

lan - Изменения в контексте 'netsh lan'.

mbn - Изменения в контексте 'netsh mbn'.

namespace - Изменения в контексте 'netsh namespace'.

nap - Изменения в контексте 'netsh nap'.

netio - Изменения в контексте 'netsh netio'.

p2p - Изменения в контексте 'netsh p2p'.
ras - Изменения в контексте 'netsh ras'.
rpc - Изменения в контексте 'netsh rpc'.
set - Обновление параметров конфигурации.
show - Отображение информации.
trace - Изменения в контексте 'netsh trace'.
wcn - Изменения в контексте 'netsh wcn'.
wfp - Изменения в контексте 'netsh wfp'.
winhttp - Изменения в контексте 'netsh winhttp'.
winsock - Изменения в контексте 'netsh winsock'.
wlan - Изменения в контексте 'netsh wlan'.

Чтобы получить справку по команде, введите эту команду, затем пробел и "?".

Также доступны следующие дочерние контексты:

advfirewall branchcache bridge dhcpclient dnsclient firewall http interface ipsec lan mbn
namespace nap netio p2p ras rpc trace wcn wfp winhttp winsock wlan

Примеры:

- netsh advfirewall show global последовательно выполняется команда первого уровня advfirewall, в ее контексте, команда следующего уровня show с параметром global.
- Как получить справку в виде текстового файла для выбранного контекста NETSH

Для примера, нужно получить справку в контексте работы с конфигурацией беспроводной сети **wlan** . Последовательно выполняем команды:

```
netsh
wlan
set file open C:\wlanhelp.txt
?
set file close
```

В данном примере, команда **set file open C:\wlanhelp.txt** устанавливает режим вывода консольных сообщений в файл с именем C:\wlanhelp.txt. После установки данного режима, все, что вводится с клавиатуры и отображается на экране, будет записано в указанный текстовый файл. Таким образом, можно создавать файлы журналов отдельных сессий использования netsh . Вместо параметра **open** можно использовать **append** и имя уже существующего файла журнала. В таком режиме данные будут записываться в конец существующего текстового файла.

- Как сохранить и восстановить сетевую конфигурацию

Команда **dump** создает сценарий, который содержит текущую конфигурацию. Если данные сценария сохранить в текстовый файл, то при необходимости, его можно будет использовать для восстановления измененных параметров с помощью команды загрузки и выполнения скриптов **exec**.

Для сохранения используется команда:

dump Имя файла сценария

Для восстановления настроек из файла сценария используется команда:

exec Имя файла сценария

В некоторых версиях netsh команда dump с указанием имени файла почему-то не работает. Однако, для сохранения конфигурации можно воспользоваться способом, описанным выше - использовать запись в файл командой **set file open C:\mynet.sav** .

netsh

set file open C:\mynet.sav

dump

quit

Остается только слегка исправить полученный файл сценария C:\mynet.sav - удалить 1-ю строчку с командой **dump** и последние - с приглашением netsh и (или) командой **quit**.

Второй способ - использовать netsh с перенаправлением вывода команды **dump** в файл:

netsh dump > C:\mynet.sav

Для сохранения отдельного контекста конфигурации можно воспользоваться командой dump на соответствующем уровне:

netsh interface dump > C:\myinterf.cnf - сохранить настройки сетевых интерфейсов в виде сценария netsh в файле C:\myinterf.cnf.

Для восстановления сетевой конфигурации можно воспользоваться

netsh exec C:\mynet.sav

Обычно, после восстановления сетевых настроек из файла сценария , требуется перезапуск некоторых сетевых служб, а желательнее - выполнить перезагрузку Windows.

- Как выполнить переключение между контекстами netsh

Иногда требуется выполнить некоторые команды на одном уровне, перейти на другой, и снова вернуться на предыдущий. Для выполнения таких переходов используются команды **pushd** и **popd** . Принцип переключения между контекстами основан на обработке очереди в соответствии с правилом "первым вошел - последним вышел" или first-in-last-out (FILO) stack. Команда **pushd** запоминает текущий уровень (контекст) в стеке, а команда **popd** извлекает его из стека. Например:

netsh> - приглашение первого уровня команды nesh

pushd - введена команда запоминания контекста в стек

netsh> - приглашении netsh не меняется, контекст прежний.

interface ipv4 - переход на уровень interface и уровень ipv4

netsh interface ipv4> - соответственно, изменилась строка приглашения, отображая текущий контекст выполнения команды netsh

set address local static 192.168.1.9 255.255.255.0 192.168.1.1 1 - команда, меняющая настройки IP протокола.

netsh interface ip> - контекст выполнения команды, отображаемый в приглашении не изменяется.

popd - команда извлечения из стека запомненного контекста.

netsh > - строка приглашения изменилась, отображая текущий контекст выполнения команды **netsh**.

Без использования команд **pushd** и **popd** практически невозможно полноценное использование сценариев **netsh**.

- Как найти примеры выполнения сетевых настроек с помощью **netsh**

Кроме сохранения и восстановления настроек использование команды **dump** позволяет получить примеры в виде сценария, соответствующего текущей конфигурации. Например, дамп секции **interface** дает пример выполнения команд **netsh** в контексте настроек сетевых интерфейсов. Пример сценария:

```
#=====
# Конфигурация интерфейса
#=====
pushd interface
reset all
popd
# Конец конфигурации интерфейса
...

# -----
# Настройка IP-интерфейсов
# -----
pushd interface ip
# Интерфейс настройки IP для "Подключение по локальной сети"

set address name=" Подключение по локальной сети " source=static addr=192.168.0.1
mask=255.255.255.0
set dns name="Подключение по локальной сети" source=static addr=192.168.0.2
mask=255.255.255.0
set wins name=" Подключение по локальной сети " source=static addr=192.168.0.9
```

Строки сценария, начинающиеся с символа **#**, являются комментариями. Команды **pushd** и **popd** позволяют определить контекст исполнения других команд **netsh**. Команды настроек конфигурации плюс справочная информация самой **netsh** позволяют довольно легко получить командную строку для выполнения отдельных сетевых настроек:

Пример: Сменить IP-адрес в командной строке:

```
netsh interface ip set address name="Подключение по локальной сети" source=static
addr=192.168.0.58 mask=255.255.255.0
```

name - имя сетевого подключения

source - static - статический IP-адрес. Возможно значение DHCP, если адрес назначается автоматически сервером DHCP.

addr - значение IP-адреса

mask - значение маски сети.

Для получения сведений о дополнительных возможностях конфигурирования сетевых интерфейсов можно перейти на соответствующий контекст выполнения netsh, и выполнить интересующую команду с параметром **?**. Например:

netsh - старт NETSH

interface - перейти в контекст настройки сетевых интерфейсов **interface**

ip - перейти в контекст настройки протокола IP

set file open C:\setaddr.txt - записывать сессию в файл. Эта команда используется, если нужна справочная информация в виде текстового файла.

set address ? выдать справку по использованию **set address**

set file close - закрыть файл справки.

quit - завершить работу с netsh

Для Windows Vista / Windows 7 синтаксис будет немного отличаться, уровню **ip** будет соответствовать уровень **ipv4**:

netsh - старт NETSH

interface - перейти в контекст настройки сетевых интерфейсов **interface**

ipv4 - перейти в контекст настройки протокола IP

set file open C:\setaddr.txt - записывать сессию в файл. Эта команда используется, если нужна справочная информация в виде текстового файла.

set address ? выдать справку по использованию **set address**

set file close

quit - завершить работу с netsh

Пример синтаксиса для смены адреса DNS-сервера в настройках сетевого подключения "Подключение по локальной сети 2" на адрес публичного DNS-сервера Google в среде Windows:

```
netsh interface ipv4 set dnsservers name="Подключение по локальной сети 2" static 8.8.8.8 primary
```

Из информации файла справки следует, что возможно использование параметров командной строки netsh без указания ключевых слов:

```
netsh interface ip set address name="Подключение по локальной сети" source=static addr=192.168.0.58 mask=255.255.255.0 gateway=192.168.0.1 gwmetric=1
```

Аналогично, без указания ключевых слов:

```
netsh interface ip set address name="Подключение по локальной сети" static 192.168.0.58 255.255.255.0 192.168.0.1 1
```

При изменении одного из параметров настроек необходимо указывать и остальные. Например, только для изменения адреса шлюза по умолчанию недостаточно выполнить команду:

netsh interface ip set address name="Подключение по локальной сети" gateway=192.168.0.1 gwmetric=1

При ее выполнении отсутствующие параметры (IP-адрес и маска) будут сброшены. Для правильной смены шлюза по умолчанию команда должна быть следующей:

netsh interface ip set address name="Подключение по локальной сети" source=static addr=192.168.0.58 mask=255.255.255.0 gateway=192.168.0.1 gwmetric=1

12. Утилита NET.EXE

Утилита NET.EXE существует во всех версиях Windows и является одной из самых используемых в практической работе с сетевыми ресурсами. Позволяет подключать и отключать сетевые диски, запускать и останавливать системные службы, добавлять и удалять пользователей, управлять совместно используемыми ресурсами, устанавливать системное время, отображать статистические и справочные данные об использовании ресурсов и многое другое.

Выполнение команды **net** без параметров вызывает краткую справку со списком возможных уровней использования, запуск с параметром **help** позволяет получить более подробную информацию об использовании **net.exe**:

Синтаксис:

NET HELP имя_команды
-или-
NET имя_команды /HELP

Параметры:

Справочная система NET.EXE, пожалуй, является одной из лучших в семействе операционных систем Windows. Подробную справку по использованию нужной команды, например **use**, можно получить несколькими способами:

net use ? - справка о синтаксисе команды

net use /help - подробная справка по использованию команды с описанием используемых ключей.

net help use - аналогично предыдущей форме вызова справки.

net help use | more - отобразить справку в постраничном режиме выдачи на экран. Удобно пользоваться в тех случаях, когда текст не помещается на экране. Нажатие **Enter** перемещает текст на одну строку, нажатие пробела - на один экран.

net help use > C:\helpuse.txt - создать текстовый файл справки C:\helpuse.txt

Команды:

NET ACCOUNTS NET HELP NET SHARE
NET COMPUTER NET HELPMMSG NET START
NET CONFIG NET LOCALGROUP NET STATISTICS
NET CONFIG SERVER NET NAME NET STOP
NET CONFIG WORKSTATION NET PAUSE NET TIME
NET CONTINUE NET PRINT NET USE
NET FILE NET SEND NET USER
NET GROUP NET SESSION NET VIEW

NET HELP SERVICES - эта команда выводит список служб, которые можно запустить.

NET HELP SYNTAX - эта команда выводит объяснения синтаксических правил, используемых при описании команд в Справке.

NET HELP имя_команды | MORE - просмотр справки по одному экрану за раз.

При описании команды NET используются следующие синтаксические соглашения:

- Заглавными буквами набраны слова, которые должны быть введены без изменений, строчными буквами набраны имена и параметры, которые могут изменяться, например, имена файлов.

- Необязательные параметры заключены в квадратные скобки - [].

- Списки допустимых параметров заключены в фигурные скобки - { }. Необходимо использовать один из элементов такого списка.

- Символ | (вертикальная черта) используется в качестве разделителя элементов списка. Возможно использование только одного из элементов списка. Например, в соответствии с изложенными соглашениями, необходимо ввести NET COMMAND и один из переключателей - SWITCH1 или SWITCH2. Указанное в квадратных скобках имя [name] является необязательным параметром:

NET COMMAND [name] { SWITCH1 | SWITCH2 }

- Запись [...] означает, что указанный элемент может повторяться. Повторяющиеся элементы должны быть разделены пробелом.

- Запись [...] означает, что указанный элемент может повторяться, но повторяющиеся элементы должны быть разделены запятой или точкой с запятой, но не пробелом.

- При вводе в командной строке можно использовать русские названия служб, при этом они должны быть заключены в кавычки и не допускается изменение прописных букв на строчные и наоборот. Например, команда

NET START "Обозреватель сети"

запускает службу обозревателя сети.

Примеры:

- Работа с системными службами

Данный режим использования **NET.EXE**, в некоторой степени, является не характерным для основного предназначения утилиты, и начиная с Windows XP, для управления системными службами используется специальная утилита командной строки **SC.EXE**. Тем не менее, NET.EXE в среде любой версии операционных систем Windows может быть использована для запуска и остановки системных служб (сервисов). Согласно справочной информации, список служб, которыми можно управлять с помощью **net.exe** можно получить используя следующую команду:

net help services

Но это не совсем верно, и на самом деле, с помощью **net.exe** можно запустить или остановить практически любую системную службу, и в том числе, не представленную в списке, отображаемом при выполнении данной команды. Для остановки используется параметр **stop**, а для запуска - параметр **start**:

net stop dnscache - остановить службу **dnscache**

net start dnscache - запустить службу **dnscache**

Возможно использование как короткого, так и полного имени ("Dnscache" - короткое, "DNS-клиент" - полное имя службы). Имя службы, содержащее символы русского алфавита и пробелы заключается в двойные кавычки.

net stop "DNS-клиент" - остановить службу **DNS-клиент** .

Полное имя службы можно скопировать из "Панель управления" - "Администрирование" - "Службы" - Имя службы - "Свойства" - "Выводимое имя". Для приостановки некоторых системных служб или продолжения работы ранее приостановленной службы используются команды **NET PAUSE** и **NET CONTINUE** :

net pause "Планировщик заданий" - приостановить службу "Планировщик заданий"
net continue schedule - продолжить работу службы "Планировщик заданий". Имя службы задано в коротком формате.

- Работа с сетевыми дисками

net use - отобразить список сетевых дисков, подключенных на данном компьютере.

Состояние	Локальный	Удаленный	Сеть
Отсоединен	X:	\\SERVER\movies	Microsoft Windows Network
OK	Y:	\\SERVER\shares	Microsoft Windows Network

В колонке "Локальный" отображается буква сетевого диска, а в колонке "Удаленный" - имя удаленного сетевого ресурса в формате **UNC**. **UNC** - это Общее соглашение об именах (Uniform Naming Convention) или универсальное соглашение об именовании (universal naming convention), соглашение об именовании файлов и других ресурсов, дающее определение местоположения ресурса. Имя, соответствующее **UNC** - полное имя ресурса в сети, включающее имя сервера и имя совместно используемого (разделяемого, сетевого) ресурса (принтера, каталога или файла). Синтаксис **UNC**-пути к каталогу или файлу следующий:

\\Сервер\СетевойКаталог[\\ОтносительныйПуть]

Сервер - сетевое имя компьютера, **СетевойКаталог** - это сетевое имя общего каталога на этом компьютере, а необязательный **ОтносительныйПуть** - путь к каталогу или файлу из общего каталога.

СетевойКаталог не обязательно называется так же, как ассоциированный с ним каталог на сервере, имя даётся в ходе открытия общего доступа к каталогу в файловой системе компьютера.

В операционных системах семейства Windows, если в конце имени разделяемого ресурса используется знак \$ то такой ресурс является скрытым и не отображается в проводнике при просмотре сетевого окружения. Это правило относится не только к автоматически создаваемым ресурсам для системного администрирования (C\$, D\$, ADMIN\$ и т.п.), но и для любого пользовательского разделяемого ресурса. Если, например, для сетевого доступа выделена папка под именем "movies", то она будет видна в сетевом окружении, а если - под именем "movies\$" - то нет. Для того чтобы скрыть в сетевом окружении отдельный компьютер используется команда:

NET config server /hidden:yes

Чтобы вернуть отображение компьютера в сетевом окружении

NET config server /hidden:no

UNC-пути можно использовать и для локальной машины, только в этом случае вместо имени "Сервер" нужно подставлять знак "?" или ".", а путь к файлу указывать вместе с буквой диска. Например так: "\\?\C:\Windows\System32\file.exe" .

Для отключения сетевого диска или устройства используется команда **net use** с ключом **/DELETE**

net use X: /delete - отключить сетевой диск X:

Регистр букв в данном ключе не имеет значения и можно использовать сокращения:

net use Y: /del

Примеры выполнения команды **NET USE** для подключения сетевых дисков:

net use X: \\server\shares - подключить сетевой диск X: которому соответствует разделяемый сетевой каталог с именем **shares** на компьютере с именем **server**

net use Y:\C\$ /USER:Администратор admpass - подключить сетевой диск Y: которому соответствует скрытый ресурс C\$ (корневой каталог диска C:) . При подключении к удаленному компьютеру используется имя пользователя **Администратор** и пароль **admpass**

То же самое, но с использованием учетной записи в домене **mydomain net use Y:\C\$ /USER:mydomain\Администратор admpass**

net use Y:\C\$ /USER:Администратор@mydomain admpass

Если в командной строке пароль не задан, то он будет запрошен при подключении к сетевому ресурсу. Если ключ **/USER** не задан, то для авторизации на удаленном компьютере используется текущая учетная запись.

net use Y:\C\$ /SAVECRED - выполнить подключение с запоминанием полномочий (credentials) пользователя. При первом подключении, будет выдан запрос на ввод имени пользователя и пароля , которые будут запомнены и не будут запрашиваться при последующих подключениях. Параметр **/savecred** не работает в версиях Домашняя и Начальная Windows 7 / Windpws XP.

Для изменения режима запоминания подключенных сетевых дисков используется ключ **/PERSISTENT**

net use /PERSISTENT:NO - не запоминать сетевые подключения.

net use /PERSISTENT:YES - запоминать сетевые подключения.

Необходимо учитывать, что режим, определяемый значением ключа **/PERSISTENT**, относится к вновь создаваемым подключениям. Если, например, сетевой диск X: был создан при установленном режиме запоминания (**PERSISTENT:YES**), а затем вы выполнили смену режима командой **net use /PERSISTENT:NO** и подключили сетевой диск Y: , то после перезагрузки системы, не будет восстановлено подключение диска Y: , но будет восстановлено подключение диска X.

- Работа с файлами и каталогами

NET SHARE - эта команда позволяет выделить ресурсы системы для сетевого доступа . При запуске без других параметров, выводит информацию обо всех ресурсах данного компьютера, которые могут быть совместно использованы . Для каждого ресурса выводится имя устройства или путь и соответствующий комментарий.

net share - получить список разделяемых в локальной сети ресурсов данного компьютера. Пример списка:

Общее имя	Ресурс	Заметки
G\$	G:\	Стандартный общий ресурс
E\$	E:\	Стандартный общий ресурс
IPC\$		Удаленный IPC
ADMIN\$	C:\WINDOWS	Удаленный Admin
INSTALL	C:\INSTALL	Дистрибутивы и обновления

net share INSTALL - получить информацию о разделяемом ресурсе с именем INSTALL .

Имя общего ресурса	INSTALL
Путь	C:\INSTALL
Заметки	Дистрибутивы и обновления
Макс. число пользователей	Не ограничен
Пользователи	Administrator
Кэширование	Вручную

Для добавления нового разделяемого по сети ресурса используется параметр **/ADD**

net share TEMP="C:\Documents And Settings\LocalSettings\games" - добавить новый разделяемый каталог под именем **TEMP**

net share TEMP="C:\Documents And Settings\LocalSettings\games" /users:5 - добавить новый разделяемый каталог под именем **TEMP** с максимальным числом одновременно подключающихся пользователей равным 5 .

Кроме этого, при создании разделяемого ресурса можно указать краткое его описание (заметку) с помощью параметра **/REMARK** и режим кэширования файлов с помощью параметра **/CACHE** .

**NET SHARE имя_ресурса=диск:путь [/USERS:число | /UNLIMITED]
[/REMARK:"текст"] [/CACHE:Manual | Automatic | No] [/CACHE:Manual | Documents| Programs | None]**

Для удаления существующего разделяемого ресурса используется параметр **/DELETE**:

net share TEMP /DELETE - удалить разделяемый ресурс под именем **TEMP**

Удаление выполняется только для имени разделяемого ресурса и не затрагивает каталог локального диска, связанный с данным именем.

Для работы с файлами, открытыми по сети на данном компьютере, используется команда **NET FILE** . По каждому открытому ресурсу выводится идентификационный номер, путь файла,

имя пользователя, которым используется файл, и количество блокировок при совместном использовании. Кроме того, команда **NET FILE** позволяет закрыть совместно используемый файл и снять блокировки .

net file - получить список открытых по сети файлов .

net file 4050 /close - принудительно закрыть файл, идентификатор которого равен 4050

Для получения списка компьютеров рабочей группы или домена с разделяемыми ресурсами используется команда

net view - отобразить список компьютеров в сетевом окружении.

net view | more - отобразить список компьютеров в постраничном режиме вывода на экран.

net view > C:\computers.txt - отобразить список компьютеров с записью результатов в текстовый файл.

Синтаксис данной команды:

NET VIEW [\\имя_компьютера [/CACHE] | /DOMAIN[:имя_домена]]
NET VIEW /NETWORK:NW [\\имя_компьютера]

net view \\server - отобразить список сетевых ресурсов компьютера server

net view /DOMAIN:mydomain - отобразить список компьютеров с разделяемыми ресурсами в домене **mydomain** Если имя домена не указано, то выводится список всех доступных компьютеров локальной сети.

net view /NETWORK:NW - отобразить список серверов Novell Netware, доступных в данной локальной сети.

net view /NETWORK:NW \\NWServer - отобразить списков сетевых ресурсов сервера Netware с именем NWServer .

- Работа с пользователями и компьютерами

Утилита NET.EXE позволяет отобразить данные об учетных записях пользователей и групп, добавлять новые записи, удалять существующие, отображать параметры безопасности, связанные с авторизацией пользователей и некоторые другие операции по администрированию на локальном компьютере или контроллере домена.

NET ACCOUNTS - эта команда используется для обновления базы данных регистрационных записей и изменения параметров входа в сеть (LOGON) . При использовании этой команды без указания параметров, выводятся текущие значения параметров, определяющих требования к паролям и входу в сеть, - время принудительного завершения сессии, минимальную длину пароля, максимальное и минимальное время действия пароля и его уникальность.

Синтаксис данной команды:

NET ACCOUNTS [/FORCELOGOFF:{минуты | NO}] [/MINPWLEN:длина]

[/MAXPWAGE:{дни | UNLIMITED}] [/MINPWAGE:дни] [/UNIQUEPW:число]
[/DOMAIN]

Пример отображаемой информации по команде NET ACCOUNTS:

Принудительный выход по истечении времени через: Никогда
Минимальный срок действия пароля (дней): 0
Максимальный срок действия пароля (дней): 42
Минимальная длина пароля: 0
Хранение неповторяющихся паролей: Нет
Блокировка после ошибок ввода пароля: Никогда
Длительность блокировки (минут): 30
Сброс счетчика блокировок через (минут): 30
Роль компьютера: РАБОЧАЯ СТАНЦИЯ

При использовании в локальной сети, каждый компьютер может выполнять как роль сервера (server), предоставляющего свои ресурсы для совместного использования, так и рабочей станции (workstation), использующей разделяемые сетевые ресурсы. Основные настройки сетевых служб сервера и рабочих станций можно отобразить с помощью команд:

net config server - настройки сетевых служб для роли сервера.

net config workstation - настройки сетевых служб для роли рабочей станции.

Настройки служб сервера можно изменить с использованием параметров:

/AUTODISCONNECT:минуты - максимальное время, в течение которого сеанс пользователя может быть не активен, прежде чем соединение будет отключено. Можно использовать значение -1, которое означает, что отключение вообще не производится. Допустимый диапазон значений: от -1 до 65535; по умолчанию используется 15.

/SRVCOMMENT:"текст"

Добавляет текст комментария для сервера, который отображается на экране Windows и при выполнении команды NET VIEW. Максимальная длина этого текста составляет 48 знаков. Текст должен быть заключен в кавычки.

/HIDDEN:{ YES | NO }

Указывает, должно ли выводиться имя данного сервера в списке серверов. Учтите, что "скрытие" сервера не изменяет параметров доступа к этому серверу. По умолчанию используется значение NO.

net config server /SRVCOMMENT:"Игровой сервер" /AUTODISCONNECT:5 - автоотключение при неактивности пользователя - 5 минут..

net config server /HIDDEN:YES>/AUTODISCONNECT:-1 - автоотключение при неактивности пользователя не выполняется, сервер не отображается в сетевом окружении.

При выполнении на контроллере домена, утилита net.exe позволяет добавлять новые компьютеры в базу данных Active Directory (AD) или удалять существующие компьютеры из нее.

net computer \\notebook /add - добавить в домен компьютер notebook .

net computer \\notebook /del - удалить из домена компьютер notebook .

Для просмотра списка групп пользователей и изменения их состава, а также добавления новых или удаления существующих групп используются команды **NET GROUP** и **NET LOCALGROUP**. Первая из них используется только на контроллерах домена и предназначена для работы с группами пользователей в домене.

net group - отобразить список групп пользователей в текущем домене.

net localgroup - отобразить список групп пользователей данного компьютера. Синтаксис и назначение параметров этих команд практически не отличаются.

NET LOCALGROUP [имя_группы [/COMMENT:"текст"]] [/DOMAIN] имя_группы {/ADD /COMMENT:"текст" | /DELETE} [/DOMAIN] имя_группы имя [...] {/ADD | /DELETE} [/DOMAIN]

имя_группы - имя локальной группы, которую необходимо добавить, изменить или удалить. Если указать только имя группы, то будет выведен список пользователей или глобальных групп, являющихся членами этой локальной группы.

/COMMENT:"текст" - комментарий для новой или существующей группы. Текст должен быть заключен в кавычки.

/DOMAIN - Команда выполняется на основном контроллере домена в текущем домене. В противном случае операция выполняется на локальном компьютере.

имя [...] - Список из одного или нескольких имен пользователей, которые необходимо добавить или удалить из локальной группы. Имена разделяются пробелом. Эти имена могут быть именами пользователей или глобальных групп, но не именами других локальных групп. Если пользователь зарегистрирован в другом домене, его имени должно предшествовать имя домена (например, SALES\RALPHR).

/ADD - Добавляет имя группы или имя пользователя в локальную группу. Регистрационная запись для добавляемых пользователей или глобальных групп должна быть создана заранее.

/DELETE - Удаляет имя группы или пользователя из локальной группы.

net localgroup Администраторы - отобразить список пользователей локальной группы **Администраторы** данного компьютера.

net localgroup Администраторы testuser /add - добавление в группу **Администраторы** нового пользователя с именем **testuser**

net localgroup Администраторы testuser /delete - удалить пользователя **testuser** из группы **Администраторы** .

Для работы с учетными записями пользователей используется команда **net user**

NET USER [имя_пользователя [пароль | *] [параметры]] [/DOMAIN] имя_пользователя {пароль | *} /ADD [параметры] [/DOMAIN] имя_пользователя [/DELETE] [/DOMAIN]

имя_пользователя - имя пользователя, которое необходимо добавить, удалить, изменить или вывести на экран. Длина имени пользователя не должна превосходить 20 знаков.

пароль - пароль для учетной записи пользователя. Пароль должен отвечать установленным требованиям на длину - быть не короче, чем значение, установленное параметром **/MINPWLEN** в команде **NET ACCOUNTS**, и в то же время не длиннее 14 знаков.

***** - Вызывает открытие специальной строки ввода пароля. Пароль не выводится на экран во время его ввода в этой строке.

/DOMAIN команда будет выполняться на контроллере домена в текущем домене.

/ADD - добавление нового пользователя.

/DELETE - удаление пользователя.

Параметры - Допустимые параметры:

/ACTIVE:{YES | NO} - Активизирует учетную запись или делает ее не активной. Если учетная запись не активна, пользователь не может получить доступ к серверу. По умолчанию используется значение YES (т.е. учетная запись активна).

/COMMENT:"текст" - Добавляет описательный комментарий об учетной записи (длиной не более 48 знаков). Текст должен быть заключен в кавычки.

/COUNTRYCODE:nnn - Использует кодовую страницу нужного языка для вывода справки и сообщений об ошибках. Значение 0 означает выбор кодовой страницы по умолчанию.

/EXPIRES:{дата | NEVER} - Устанавливает дату истечения срока действия учетной записи. Если используется значение NEVER, то время действия учетной записи не ограничено. Дата истечения срока действия задается в формате дд/мм/гг или мм/дд/гг, в зависимости от того, какая кодовая страница используется. Месяц может быть указан цифрами, названием месяца или трехбуквенным его сокращением. В качестве разделителя полей должен использоваться знак косой черты (/).

/FULLNAME:"имя" - Указывает настоящее имя пользователя (а не кодовое имя, заданное параметром имя_пользователя). Настоящее имя следует заключить в кавычки.

/HOMEDIR:путь Указывает путь к домашнему каталогу пользователя. Этот каталог должен существовать.

/PASSWORDCHG:{YES | NO} Определяет, может ли пользователь изменять свой пароль. По умолчанию используется значение YES (т.е. изменение пароля разрешено).

/PASSWORDREQ:{YES | NO} Определяет, является ли указание пароля обязательным. По умолчанию используется значение YES (т.е. пароль обязателен).

/PROFILEPATH[:путь] Устанавливает путь к профилю пользователя.

/SCRIPTPATH:путь Устанавливает расположение пользовательского сценария для входа в систему.

/TIMES:{промежуток | ALL} - Устанавливает промежуток времени, во время которого пользователю разрешен вход в систему. Этот параметр задается в следующем формате: день[-день][,день[-день]],время[-время][,время[-время]]

Время указывается с точностью до одного часа. Дни являются днями недели и могут указываться как в полном, так и в сокращенном виде. Время можно указывать в 12- и 24-часовом формате. Если используется 12-часовой формат, то можно использовать am, pm, a.m. или p.m. Значение ALL указывает, что пользователь может войти в систему в любое время, а пустое значение указывает, что пользователь не может войти в систему никогда. Разделителем полей указания дней недели и времени является запятая, разделителем при использовании нескольких частей является точка с запятой.

/USERCOMMENT:"текст" - Позволяет администратору добавлять или изменять текст комментария к учетной записи. **/WORKSTATIONS:{имя_компьютера[,...] | *} -** Перечисляет до восьми различных компьютеров, с которых пользователь может войти в сеть. Если данный параметр имеет пустой список или указано значение *, пользователь может войти в сеть с любого компьютера.

Примеры использования:

net user - отобразить список пользователей

net user /DOMAIN - отобразить список пользователей текущего домена

net user VASYA /USERCOMMENT:"Тестовый пользователь " /add - добавить пользователя с именем VASYA

net user VASYA /delete - удалить созданного пользователя.

net user VASYA password /USERCOMMENT:"Тестовый пользователь " /add - создать учетную запись нового пользователя VASYA с паролем password .

net user VASYA * /USERCOMMENT:"Тестовый пользователь " /add - то же, что и в предыдущей команде, но пароль будет запрошен при создании новой учетной записи.

net user VASYA * - изменить пароль существующего пользователя VASYA. Новый пароль будет запрошен при выполнении команды.

net user VASYA Boss - изменить пароль пользователя VASYA на новое значение Boss

Пример последовательности команд для создания нового пользователя с правами локального администратора:BR>

net user VASYA Boss /ADD - создание учетной записи.

net localgroup Администраторы VASYA /ADD - добавление пользователя в группу "Администраторы"

- Отправка сообщений по локальной сети

Для отправки сообщений в локальной сети используется команда **NET SEND**

NET SEND {имя | * | /DOMAIN[:имя] | /USERS} сообщение имя - имя пользователя, компьютера или имя для получения сообщений, на которое отправляется данное сообщение. Если это имя содержит пробелы, то оно должно быть заключено в кавычки (" ").

***** - отправка сообщения по всем именам, которые доступны в данный момент.

/DOMAIN[:имя домена] - сообщение будет отправлено по всем именам домена данной рабочей станции. Если указано имя домена, то сообщение отправляется по всем именам указанного домена или рабочей группы.

/USERS - сообщение будет отправлено всем пользователям, подключенным в настоящий момент к серверу.

сообщение - текст отправляемого сообщения.

Для того, чтобы получить сообщение, должна быть запущена "Служба сообщений" (MESSENGER). Имена пользователей, компьютеров и текст сообщений на русском языке должны быть в DOS-кодировке.

Перечень доступных активных имен на данном компьютере и состояние службы сообщений можно получить с использованием команды **net name** без параметров. По всему списку имен, отображаемому в результате выполнения данной команды возможна отправка сообщений.

Примеры использования:

net send VASYA привет! - отправка сообщения на имя VASYA .

net send * привет! - отправка сообщения всем пользователям локальной сети, имена которых можно определить.

net send /DOMAIN:mydomain Привет - отправка сообщения всем пользователям в домене mydomain

net send /USERS Привет! - отправка сообщений всем пользователям, зарегистрированным службой сервера данного компьютера.

- Статистика и синхронизация часов

Утилита NET.EXE позволяет получить статистические данные по использованию служб сервера и рабочей станции. Статистика содержит информацию о сеансах, доступе к сетевым устройствам, объемах принятых и переданных данных, отказах в доступе и ошибках, обнаруженных в процессе сетевого обмена.

net statistics server - отобразить статистические данные для службы сервера

net statistics workstation - отобразить статистические данные для службы рабочей станции

Для изменения системного времени компьютера используется команда **NET TIME**:

NET TIME [\\компьютер | /DOMAIN[:домен]] /RTSDOMAIN[:домен]] [/SET]
[\\компьютер] /QUERYSNTP [\\компьютер] /SETSNTp[:список серверов NTP]

NET TIME синхронизирует показания часов компьютера с другим компьютером или доменом. Если используется без параметров в домене Windows Server, выводит текущую дату и время дня, установленные на компьютере, который назначен сервером времени для данного домена. Эта команда позволяет задать сервер времени NTP для компьютера. \\компьютер - имя компьютера, который нужно проверить или с которым нужно синхронизировать показания часов.

/DOMAIN[:домен] Задаёт домен, с которым нужно синхронизировать показания часов.

/RTSDOMAIN[:домен] - выполняет синхронизацию времени с сервером времени (Reliable Time Server) из указанного домена.

/SET - Синхронизирует показания часов компьютера со временем указанного компьютера или домена.

/QUERYSNTP - Отображает назначенный этому компьютеру сервер NTP /SETSNTp[:ntp server list] - задать список серверов времени NTP для этого компьютера. Это может быть список IP-адресов или DNS-имен, разделенных пробелами. Если задано несколько серверов, список должен быть заключен в кавычки.

net time \\COMPUTER - отобразить время на компьютере COMPUTER. Вместо имени компьютера можно использовать его IP-адрес.

net time \\COMPUTER /SET - установить часы текущего компьютера по значению часов компьютера COMPUTER

net time \\COMPUTER /SET /YES - установить часы текущего компьютера по значению часов компьютера COMPUTER без запроса подтверждения. Обычно ключ /YES используется в командных файлах, выполняющихся без участия пользователя.

net time /QUERYSNTP - отобразить сервер времени, определенный для данного компьютера.

net time \\COMPUTER /QUERYSNTP - отобразить сервер времени, определенный для указанного компьютера.

net time /SETSNTp:"1.ru.pool.ntp.org time.windows.com" - задать в качестве NTP-серверов узлы 1.ru.pool.ntp.org и time.windows.com

13. Утилита Nslookup.exe

Утилита NSLOOKUP присутствует во всех версиях операционных систем Windows и является классическим средством диагностики сетевых проблем, связанных с разрешением доменных имен в IP-адреса. NSLOOKUP предоставляет пользователю возможность просмотра базы данных DNS-сервера и построения определенных запросов, для поиска нужных ресурсов DNS. Практически, утилита выполняет функции службы DNS-клиент в командной строке

Windows. После запуска, утилита переходит в режим ожидания ввода. Ввод символа ? или команды **help** позволяет получить подсказку по использованию утилиты.

Примеры:

nslookup - запуск утилиты

yandex.ru. - отобразить IP-адрес (а) узла с именем yandex.ru . Точка в конце имени желательна для минимизации числа запросов на разрешение имени к серверу DNS. Если завершающей точки нет, то NSLOOKUP сначала попытается разрешить указанное имя как часть доменного имени компьютера, на котором она запущена.

server 8.8.4.4 - установить в качестве сервера имен DNS-сервер Google с IP-адресом 8.8.4.4

yandex.ru. - повторить запрос с использованием разрешения имени DNS-сервером Yandex.

set type=MX - установить тип записи MX

yandex.ru. - отобразить MX-запись для домена yandex.ru - В примере узел обмена почтой для домена - mx.yandex.ru

mx.yandex.ru. - отобразить информацию по mx.yandex.ru

set type=A - установить тип записи в A

mx.yandex.ru - получить IP-адреса для mx.yandex.ru .

exit - завершить работу с nslookup

Возможно использование утилиты NSLOOKUP не в интерактивном режиме:

nslookup odnoklassniki.ru - определить IP-адрес узла odnokassniki.ru с использованием сервера DNS, заданного настройками сетевого подключения.

nslookup odnoklassniki.ru 8.8.8.8 - определить IP-адрес узла odnokassniki.ru с использованием DNS-сервера 8.8.8.8 (публичный DNS-сервер Google)

2.3 Примеры практического использования сетевых утилит командной строки.

1. Управление профилями беспроводных сетей

Обычно, если вы хотите изменить или удалить профиль беспроводного подключения, это можно сделать, щелкнув правой кнопкой мыши сеть в списке и выбрав команду в появившемся меню. Но для выполнения некоторых задач необходимо использовать командную строку. В следующей таблице показано, как выполнять стандартные задачи.

Задача	Инструкции
Показать все профили беспроводной связи на компьютере	В командной строке введите: netsh wlan show profiles
Показать ключ безопасности доступного профиля	Нажмите и удерживайте или щелкните правой кнопкой мыши сеть в списке, а затем коснитесь или щелкните Просмотреть свойства подключения.
Показать ключ безопасности профиля вне доступа	В командной строке введите: netsh wlan show profile name="ProfileName" key=clear
Удалить доступный профиль	Нажмите и удерживайте или щелкните правой кнопкой мыши сеть в списке, а затем коснитесь или щелкните Забыть эту сеть.
Удалить профиль вне доступа	В командной строке введите: netsh wlan delete profile name="ProfileName"

Переместить сеть вверх в списке приоритета	Чтобы поместить сеть в верхнюю строку списка, достаточно подключиться к ней и установить автоподключение.
Отключить автоматическое подключение к до- ступной сети	Коснитесь или щелкните сеть в списке, затем нажмите Отключе- ние.
Отключить автоматическое подключение к сети вне доступа	В командной строке введите: netsh wlan set profileparameter name="ProfileName" connec- tionmode=manual

2. Определение подмены адреса узла в файле hosts

Одним из последствий вирусного заражения довольно часто является блокировка доступа к сайтам антивирусных компаний, поисковым системам, популярным социальным сетям (Vkontakte, Odnoklassniki, Facebook, Twitter и т.п.). Подобный же прием используется для кражи учетных данных пользователей путем перенаправления на вредоносный сайт, адрес которого берется из зараженного файла **hosts**.

Порядок преобразования доменных имен в IP-адреса следующий:

- проверяется наличие данных об имени в кэш службы разрешения имен (процедура определения IP по имени уже выполнялась, и в памяти есть актуальные результаты). Если запись есть, то будут использованы ее данные.

- проверяется наличие записи об имени и адресе в файле **hosts**. Если запись есть, то будут использованы ее данные.

- для разрешения доменного имени в IP-адрес выполняется запрос к серверу DNS, заданному в настройках сетевого подключения.

Файл **hosts** при настройках по умолчанию, находится в каталоге \Windows\system32\drivers\etc\ и обычно содержит строки, начинающиеся с символа # , являющиеся комментариями, и одну запись для определения имени узла петлевого интерфейса:

127.0.0.1 localhost

127.0.0.1 - IP-адрес, localhost - имя. Если добавить запись **127.0.0.1 odnoklassniki.ru**, то для имени odnoklassniki.ru будет использоваться адрес 127.0.0.1, который не предназначен для выполнения реальной передачи данных, и сервер с указанным именем станет недоступен. Если же вместо адреса 127.0.0.1 использовать адрес поддельного сервера, созданного злоумышленниками, то вместо реального сайта, соответствующего доменному имени, посетитель перейдет на поддельную страницу.

Структура записей файла **hosts** предполагает, что между адресом и соответствующим ему именем должен быть хотя бы один символ табуляции (пробел). Каждой записи отводится одна строка в файле **hosts**. Иногда, вредоносная программа выполняет смещение записей относительно отображаемой на экране части файла, заполняя видимую часть пробелами, а в непомещающейся в области просмотра части, могут присутствовать записи, например

```
31.214.145.172 odnoklassniki.ru
31.214.145.172 www.facebook.com
31.214.145.172 www.vk.com
31.214.145.172 www.vkontakte.ru
```

Данный адрес взят из реально зараженного файла **hosts** и принадлежит сети одного из провайдеров Германии. Сейчас он безопасен, и не занят обслуживанием вредоносного сервера. На зараженном компьютере, в файл **hosts** было добавлено множество пустых строк, и поддельные записи располагались с разным смещением относительно начала строки, что могло затруднить ручной поиск. Кроме того, вредоносные программы могут использовать и некоторые другие способы подмены содержимого **hosts** - изменение местоположения самого файла, использование атрибута "скрытый" и имени с подменой символа на похожий по написанию символ национального алфавита - "о" и т.п. Другими словами, достоверно определить сам факт подмены адреса с помощью файла **hosts**, путем прямого анализа содержимого реестра, системных каталогов и самого файла занимает довольно длительное время и не всегда позволяет исключить ошибку поиска вредоносных записей. А, тем временем, задача легко решается с использованием всего лишь 2-х команд из рассмотренных выше - **ping** и **nslookup**.

ping odnoklassniki.ru - в ответе на пинг будет отображаться адрес, соответствующий имени **odnoklassniki.ru** при определении IP-адреса на данном компьютере

nslookup odnoklassniki.ru - получить IP-адрес, соответствующий имени **odnoklassniki.ru** от сервера DNS.

Если адрес по результатам пинга отличается от адреса, полученного от DNS-сервера, то присутствует факт подмены содержимого файла **hosts**. Для некоторых крупных доменов утилита **nslookup** может выдавать список из нескольких IP. Тогда IP-адрес, полученный в результатах пинга, должен присутствовать в списке адресов от **nslookup**.

Иногда, в качестве способа блокировки определенных сайтов, используется добавление несуществующих статических маршрутов для соответствующих IP-адресов или подсетей, что легко отследить с помощью утилиты **tracert**.

3. Как открыть порт в брандмауэре Windows

Разрешить входящие соединения через брандмауэр Windows (открыть порт) можно с использованием контекста **firewall** утилиты **netsh**

netsh firewall set portopening protocol=TCP port=27015 name=MyServer mode=ENABLE scope=ALL

или

netsh firewall set portopening TCP 27015 MyServer ENABLE ALL

protocol - Протокол порта. TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ALL - Все протоколы.

port - Номер порта.

name - Имя порта (необязательно)

mode - Режим порта. ENABLE - Пропускать через брандмауэр (по умолчанию). DISABLE - Не пропускать через брандмауэр.

scope - Область порта (необязательно). ALL - Пропускать через брандмауэр весь трафик (по умолчанию). SUBNET - Пропускать через брандмауэр только трафик локальной сети (подсети). CUSTOM - Пропускать через брандмауэр только указанный трафик.

С учетом значений по умолчанию и необязательных параметров открыть TCP порт 27015 в брандмауэре Windows можно командой

netsh firewall set portopening TCP 27015

В Windows Vista/Windows7 пока поддерживается синтаксис приведенный в примере выше, однако в последующих версиях операционных систем он будет полностью заменен на контекст **netsh advfirewall** - управление улучшенным брандмауэром. Подсказку по использованию можно получить при вводе команды с параметром ? (знак вопроса):

netsh advfirewall ?

В контексте правил для брандмауэра:

netsh advfirewall firewall ?

Для открытия порта 27015 в Windows 7 с учетом нового синтаксиса правильнее использовать команду:

```
netsh advfirewall firewall add rule name="Open Port 27015" dir=in action=allow protocol=TCP localport=27015
```

add rule - добавить правило

name - название правила. Название может быть произвольным, и если текст содержит пробелы - заключаться в двойные кавычки. Имя правила не должно принимать значение **all**

dir - направление обмена данными (in-входящий трафик, out- исходящий)

action - действие по отношению к попадающему под правило соединению (allow - разрешить, block - запретить)

protocol - разновидность протокола. (TCP - протокол TCP, UDP - протокол UDP, ANY - любой протокол). Если параметр **protocol** не указан, то используется значение по умолчанию - ANY)

localport - номер порта на локальном компьютере. Можно указывать диапазон портов 0 -65535 или any - любой порт или номера через запятую - 67,69.

Примеры правил брандмауэра Windows 7

По сравнению с предыдущими версиями Windows синтаксис правил стал немного сложнее, но и возможности брандмауэра значительно расширились.

Краткий список возможных параметров правил:

add rule name=<строка>

dir=in|out

action=allow|block|bypass

[program=<путь к программе>]

[service=<краткое имя службы>|any]

[description=<строка>]

[enable=yes|no (по умолчанию - yes)]

[profile=public|private|domain|any[,...]]

[localip=any||<подсеть>|<диапазон>|<список>]

[remoteip=any|localsubnet|dns|dhcp|wins|defaultgateway| ||<подсеть>|<диапазон>|<список>]

[localport=0-65535||<диапазон портов>[,...]]RPC|RPC-EPMap|IPHTTPS|any (по умолчанию - any)]

[remoteport=0-65535|<диапазон портов>[,...]|any (по умолчанию - any)]
[protocol=0-255|icmpv4|icmpv6|icmpv4:тип,код|icmpv6:тип,код|tcp|udp|any (по умолчанию - any)]
[interfacetype=wireless|lan|ras|any]
[rmtcomputergrp=<строка SDDL>]
[rmtusrgrp=<строка SDDL>]
[edge=yes|deferapp|deferuser|no (по умолчанию - no)]
[security=authenticate|authenc|authdynenc|authnoencap|notrequired (по умолчанию – notrequired)]

Некоторые правила применения параметров:

Параметры могут следовать в произвольном порядке - **dir=in action=allow** и **action=allow dir=in** являются допустимыми значениями.

Если указана удаленная группа пользователей или компьютеров, для параметра **security** необходимо установить значение **authenticate**, **authenc**, **authdynenc** или **authnoencap**. Установка **authdynenc** в качестве значения параметра **security** позволяет системам динамически согласовывать использование шифрования трафика, соответствующего данному правилу брандмауэра Windows. Шифрование согласуется в соответствии со свойствами существующего правила безопасности соединения. Этот параметр позволяет компьютеру принять первый пакет TCP или UDP входящего соединения IPsec, при условии, что он защищен, но не зашифрован, с помощью IPsec. Как только первый пакет будет обработан, сервер повторно согласует соединение и обновит его, чтобы все последующие соединения были полностью зашифрованы.

Если **action=bypass**, должна быть указана группа удаленных компьютеров, если **dir=in**.

Короткое имя службы можно посмотреть в ее свойствах, в поле **Имя службы**. Так, для службы "DNS-клиент" короткое имя - **Dnscache**. Если **service=any**, правило действует только для служб.

Значением кода или типа ICMP может быть **any** - любой ICMP трафик.

Параметр **edge** можно указывать только для правил входящего трафика (**dir=in**).

AuthEnc и **authnoencap** нельзя использовать вместе. Если задан параметр **authnoencap**, то параметр **security=authenticate** задавать необязательно.

Параметр **Authdynenc** допустим только в том случае, если значение **dir** равно **in**.

Примеры:

Добавление правила для входящего трафика для программы qip.exe:

```
netsh advfirewall firewall add rule name="allow QIP" dir=in program="c:\program-files\qip\qip.exe" action=allow
```

Добавление правила, запрещающего исходящий трафик для TCP порта 80:

```
netsh advfirewall firewall add rule name="allow80" protocol=TCP dir=out localport=80 action=block
```

Добавление правила входящего трафика с требованием безопасности и шифрования для трафика через TCP-порт 80:

```
netsh advfirewall firewall add rule name="Require Encryption for Inbound TCP/80" protocol=TCP dir=in localport=80 security=authdynenc action=allow
```

Добавление правила входящего трафика для messenger.exe с требованием безопасности:

```
netsh advfirewall firewall add rule name="allow messenger" dir=in program="c:\program files\messenger\msmsgs.exe" security=authenticate action=allow
```

Добавление правила обхода брандмауэра с проверкой подлинности для группы asmedomain\scanners, определяемой строкой SDDL:

```
netsh advfirewall firewall add rule name="allow scanners" dir=in rmtcomputergrp=<строка SDDL> action=bypass security=authenticate
```

Добавление правила разрешения исходящего трафика для локальных портов 5000-5010 для udp:

```
netsh advfirewall firewall add rule name="Allow port range" dir=out protocol=udp localport=5000-5010 action=allow
```

Для просмотра всех правил брандмауэра используется команда:

```
netsh advfirewall firewall show rule name=all  
netsh advfirewall firewall show rule name=all | more - с выдачей результатов на экран в страничном режиме  
netsh advfirewall firewall show rule name=all > C:\firewallrules.txt - с выдачей результатов в файл
```

Для просмотра конкретного правила указывается его имя. Для удаления правила используется параметр **delete**:

```
netsh advfirewall firewall show rule name=TEST просмотр правила с именем TEST  
netsh advfirewall firewall delete rule name=test - удаление правила с именем TEST
```

Для изменения значений в существующих правилах используется параметр **set** и **new** перед изменяемым значением:

```
netsh advfirewall firewall set rule name="Allow port range" new localport=5000-6000 изменить диапазон портов для правила "Allow port range"
```

Настройками по умолчанию, в режиме повышенной безопасности брандмауэр Windows 7 блокирует все входящие подключения, не соответствующие ни одному правилу и разрешает исходящие.

4. Wi-Fi точка доступа стандартными средствами Windows 7

В операционной системе Windows 7 реализована технология **Virtual WiFi**, позволяющая легко создавать **программную точку доступа (Software Access Point - SoftAP)**. В отличие от полноценных беспроводных точек доступа, реализуемая таким образом SoftAP, позволяет создать только один виртуальный адаптер, который будет работать только в режиме точки доступа, и может быть использовано шифрование только по WPA2-PSK/AES. Тем не менее, этого вполне достаточно для создания функциональной беспроводной сети без реально существующей точки доступа. Такая сеть, обозначается как **Wireless Hosted Network**, или просто **Hosted Network (Размещенная Сеть)**.

Для создания размещенной сети используется команды сетевой оболочки **netsh.exe** в контексте **wlan**:

netsh wlan set hostednetwork [mode=]allow|disallow - разрешить или запретить использование размещенной сети.

netsh wlan set hostednetwork [ssid=]<идентификатор_SSID> [key=]<парольная_фраза> [keyUsage=]persistent|temporary - задать параметры размещенной сети.

ssid - идентификатор SSID сети, другими словами - имя беспроводной сети;

key - ключ безопасности, используемый в данной сети, т.е. парольная фраза, используемая при подключении клиентов к виртуальной точке доступа. Ключ должен быть строкой символов ASCII длиной от 8 до 63 знаков.

keyUsage - указывает, является ключ безопасности постоянным или временным. По умолчанию, ключ является постоянным (**persistent**) и используется при каждом включении размещенной сети.

Примеры:

```
set hostednetwork mode=allow
set hostednetwork ssid=ssid1
set hostednetwork key=passphrase keyUsage=persistent
set hostednetwork mode=allow ssid=MyWiFi key=MyPassWord
```

Или - одной командной строкой:

netsh wlan set hostednetwork mode=allow ssid=MyWiFi key=MyPassWord - создать виртуальную точку доступа Wi-Fi с именем **MyWiFi** и паролем **MyPassWord**

Созданная программная точка доступа не будет запущена автоматически. Для запуска потребуется выполнить команду:

netsh wlan start hostednetwork

Для остановки - **netsh wlan stop hostednetwork**

При использовании команд управления размещенной сетью требуются права администратора.

Для организации доступа в Интернет с использованием размещенной сети можно воспользоваться совместным подключением через, созданный после выполнения команды создания размещенной сети, виртуальный сетевой адаптер - **Адаптер мини-порта виртуального WiFi Microsoft (Microsoft Virtual WiFi miniport adapter)**. Если же данный адаптер не обнаруживается в диспетчере устройств и отсутствует в списке сетевых адаптеров, то наиболее вероятно, что драйвер реального Wi-Fi устройства не сертифицирован для использования в операционной системе Windows 7 и не поддерживает технологию **Virtual WiFi**.

3. Задания для выполнения

1. Запустить командную консоль.
2. Попыаться выполнить все приведенные сетевые команды с различными параметрами.
3. Исследовать, как параметры влияют на результат выполнения команд.
4. Написать программу на любом языке программирования для взаимодействия пользователя со следующими утилитами: утилита для настройки TCP/IP config, ping, tracert.

4. Варианты индивидуальных заданий.

К написанной программе (п. 3 №4) добавить следующие утилиты (возможности):

1. NETSTAT, редактирование файла hosts.
2. Настройка firewall.
3. Полная реализация утилиты ipconfig, редактирование firewall.
4. Настройка Wi-Fi точки доступа.
5. NBTSTAT, GETMAC.
6. ROUTE, NET.
7. NETSH.
8. ARP, NSLOOKUP.

ЛАБОРАТОРНАЯ РАБОТА №2 УСТАНОВКА И НАСТРОЙКА ОДНОРАНГОВОЙ СЕТИ НА ОСНОВЕ ОС WINDOWS NT/2000/XP/2003/7. ИЗУЧЕНИЕ ОСНОВНЫХ СЕТЕВЫХ УТИЛИТ.

1. Цель работы

1. Выяснить назначение виртуальных машин. Научиться использовать их для работы с операционными системами семейства Windows.
2. Знать базовые принципы IP-адресации.
3. Научиться настраивать стек протоколов TCP/IP и уметь организовывать одноранговую сеть.
4. Ознакомиться и уметь использовать основные консольные утилиты для получения информации о сетевой конфигурации системы и тестирования работоспособности сети.

2. Общие сведения

2.1 Понятие локальных вычислительных сетей

Локальная вычислительная сеть, ЛВС (англ. Local Area Network, LAN) – компьютерная сеть, покрывающая относительно небольшую территорию, такую как дом, офис, или небольшую группу зданий, например, институт.

Компьютеры могут соединяться по различным протоколам, таким как **wi-fi** или **Ethernet**. ЛВС может иметь шлюзы с другими локальными сетями; быть частью или иметь подключение к глобальной вычислительной сети, например к Интернет.

Классические архитектуры организации локальных сетей

Одноранговые, или децентрализованные сети – это компьютерные сети, основанные на равноправии участников. В таких сетях отсутствуют выделенные серверы, а каждый узел (peer) является как клиентом, так и сервером (см. рис. 1).

К основным достоинствам одноранговых сетей можно отнести:

- низкая стоимость;
- высокая надежность;
- простота реализации, настройки, сопровождения для относительно небольших проектов.

Но данной архитектуре присущи и недостатки:

- зависимость эффективности работы сети от количества станций;
- сложность управления сетью при большом количестве узлов;
- сложность обеспечения защиты информации;
- трудности обновления и изменения программного обеспечения станций.

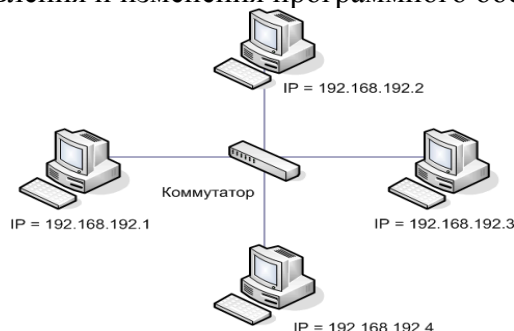


Рис. 1. Пример одноранговой сети

Сеть с выделенным сервером – это локальная вычислительная сеть (LAN), в которой сетевые устройства централизованы и управляются одним или несколькими серверами. Индивидуальные рабочие станции или клиенты (такие, как ПК) должны обращаться к ресурсам сети через сервер(ы) (см. рис. 2).

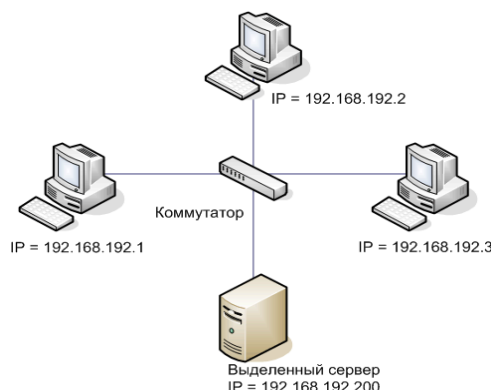


Рис. 2. Пример сеть с выделенным сервером

Достоинства сети с выделенным сервером:

- надежная система защиты информации;
- высокое быстродействие;
- отсутствие ограничений на число рабочих станций;
- простота управления по сравнению с одноранговыми сетями.

Недостатки данной архитектуры:

- более высокая стоимость из-за выделения одного компьютера под сервер;
- зависимость быстродействия и надежности сети от сервера;
- меньшая гибкость по сравнению с одноранговой сетью.

2.2 Виртуальные машины

Виртуальная машина – это вид программного обеспечения, эмулирующего работу аппаратного обеспечения компьютера. Как и в случае с реальной машиной, вы можете установить на виртуальную машину операционную систему, причем неважно Windows или *nix. Таким образом вы можете тестировать различные операционные системы не покидая своей. У виртуальной машины эмулируются BIOS, жесткий диск (на базе – файлов), CD-ROM (физический CD-привод или подключенный ISO-образ), сетевые адаптеры для соединения с вашей реальной машиной, сетевыми ресурсами или другими виртуальными машинами и т.д. Вы можете без проблем обмениваться файлами между основной операционной системой (host) и гостевой операционной системой (guest). Это осуществляется различными способами от общих («расшаренных») каталогов до простого перетаскиванием файлов из файлового менеджера клиента в окно гостевой системы или в обратном направлении.

Удобство виртуальной машины для тестирования автоматической установки просто неопределимо. Достаточно просто подключить загрузочный ISO-образ вместо CD-ROM в настройках виртуальной машины, и установка системы пойдет точно так же, как и на реальной машине.

В ходе изучаемого курса «Сети ЭВМ и телекоммуникации» мы неоднократно будем использовать возможности виртуализации для настройки, тестирования и изучения принципов работы сетевых операционных систем, протоколов и т.д.

Настройка Microsoft Virtual PC (в качестве альтернативного ПО можете использовать Oracle VM VirtualBox)

На данный момент для пользователей доступно на бесплатной основе несколько различных виртуальных машин. Наиболее популярные программы виртуализации: Microsoft Virtual PC, VmWare, Innotek VirtualBox.

В дальнейшем на занятиях предлагается использовать программу Microsoft Virtual PC 2007. Она бесплатна и очень легко настраивается. К тому же для упрощения работы на лабораторных занятиях предлагаются предустановленные виртуальные машины для ряда операционных систем. В частности, Windows 2000 SP2.

Инсталляция программы доступна на кафедральном ftp-сервере.

Основное рабочее окно программы после установки выглядит аналогично рис. 3.

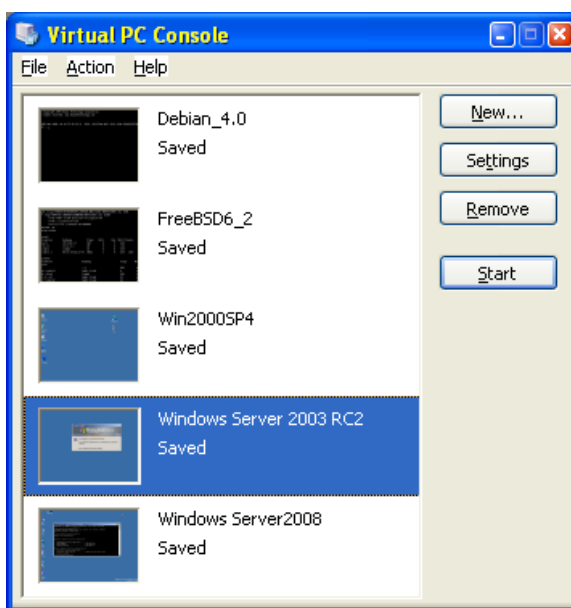


Рис. 3. Менеджер виртуальных машин

Заметьте, что сразу после установки список доступных виртуальных машин будет пуст.

Рассмотрим с вами процесс подключения и конфигурирования виртуальных машин на примере предустановленной ОС Windows 2000sp2 (образы доступны на кафедральном сервере).

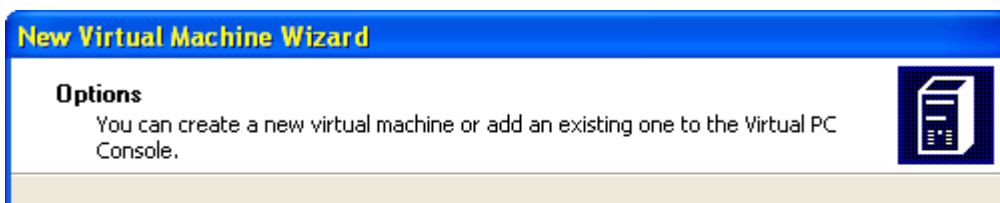
Для начала необходимо скачать на локальную машину 2 файла: *.vhd (образ жесткого диска) и *.vmc (конфигурационный файл).

Далее для подключения образа необходимо выбрать пункт меню «File->New Virtual Machine Wizard». После чего в мастере создания виртуальных машины выбрать пункт «Add an existing virtual machine» (см. рис. 4.)

После выбора соответствующего конфигурационного файла (рис. 5), машина появляется в списке доступных для запуска (рис. 6). Вы также можете изменить настройки машины (рис. 7). Например, изменить объем выделяемой оперативной памяти, добавить или удалить сетевой адаптер и т.д.

После установки и настройки виртуальной машины, ее можно запустить посредством кнопки «Start».

Работа в виртуальной среде практически **полностью** идентичная работе с host-машиной. Имеющиеся отличия заключаются в интерпретации комбинаций служебных клавиш. Информацию об этом можно легко почерпнуть из меню «File», где для каждой служебной команды сопоставлено сочетание клавиш.



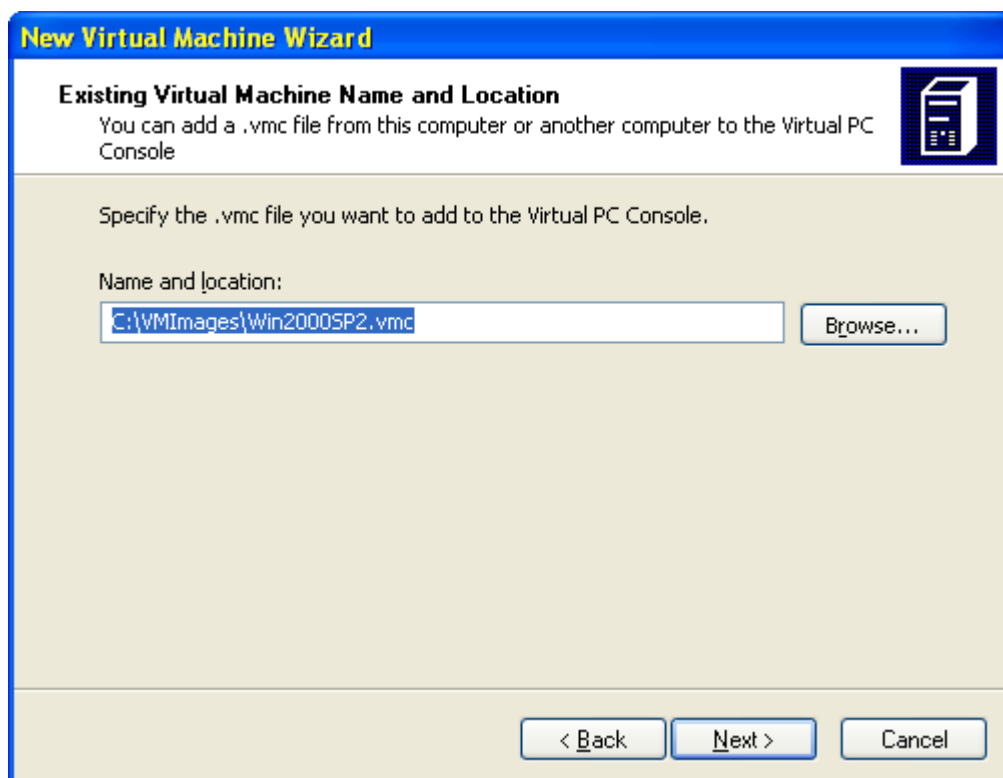
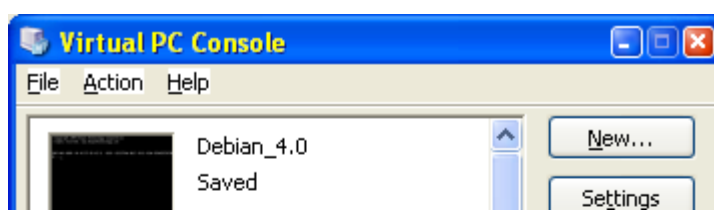


Рис. 5. Выбор конфигурационного файла



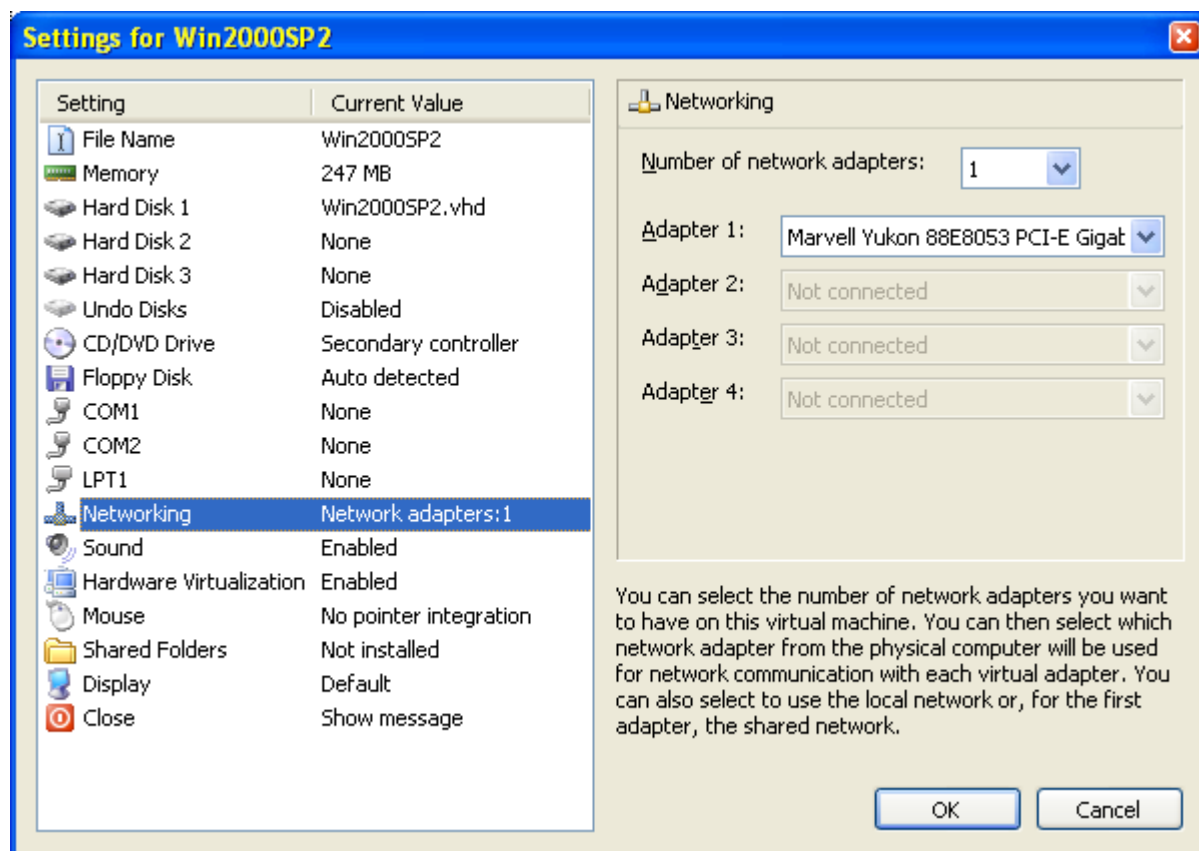


Рис. 7. Настройка виртуальной машины

2.3 Локальные сети по технологии Ethernet

На данный момент самой распространенной аппаратной технологией организации локальных сетей является Ethernet. Сейчас локальные сети преимущественно строят по звездообразной топологии с использованием сетевых карт, коммутаторов и витой пары. Далее мы рассмотрим перечисленные компоненты более подробно.

Сетевая карта (сетевой адаптер, Ethernet-адаптер, NIC (network interface card)) – печатная плата, позволяющая подключаться к компьютерной сети (см. рис. 8).

Обычно, сетевая плата идет как отдельное устройство и вставляется в слоты расширения материнской платы (в основном, PCI). На современных материнских платах, сетевой адаптер все чаще является встроенным, таким образом, покупать отдельную плату не нужно до тех пор, пока не требуется организация еще одного сетевого интерфейса.

На сетевой плате для подключения к локальной сети имеются разъемы для подключения кабеля витой пары (например, RJ-45), а также несколько информационных светодиодов, сообщающих о наличии подключения и передаче информации.



Рис. 8. Сетевой Ethernet-адаптер с разъемом RJ-45

Сетевой коммутатор, или свитч (switch - переключатель) – устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного сегмента. В отличие от концентратора (hub), который распространяет трафик от одного подключенного устройства ко всем остальным, коммутатор передает данные только непосредственно получателю. Это повышает производительность и безопасность сети, избавляя остальные сегменты сети от необходимости (и возможности) обрабатывать данные, которые им не предназначались.



Рис. 9. Пример 24-х портового коммутатора

Витая пара (англ. twisted pair) – вид кабеля связи, представляет собой одну или несколько пар изолированных проводников, скрученных между собой (с небольшим числом витков на единицу длины), для уменьшения взаимных наводок при передаче сигнала, и покрытых пластиковой оболочкой (см. рис 10). Один из компонентов современных структурированных кабельных систем. Используется в телекоммуникациях и в компьютерных сетях в качестве сетевого носителя во многих технологиях, таких как Ethernet, ARCNet и Token ring. В

настоящее время, благодаря своей дешевизне и лёгкости в установке, является самым распространённым для построения локальных сетей.

Кабель подключается к сетевым устройствам при помощи соединителя RJ-45 (см. рис. 11), немного бОльшим, чем телефонный соединитель RJ11.

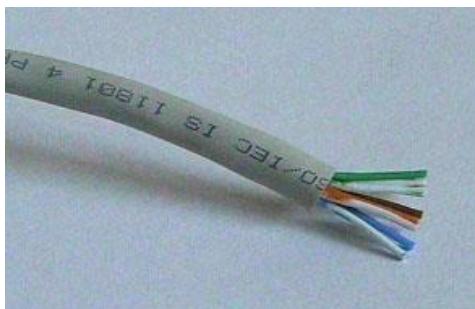


Рис. 10. Неэкранированная витая пара

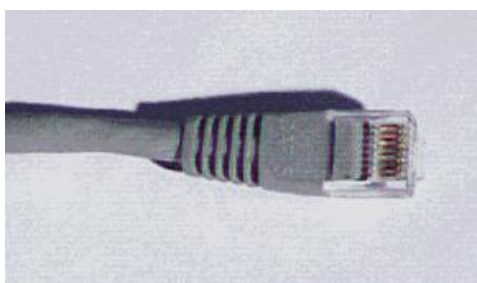


Рис. 11. Витая пара с коннектором RJ-45

2.4 Адресация в локальных сетях

Для взаимодействия узлов на аппаратном уровне каждый активный участник сети (например, сетевой адаптер) должен быть адресуем некоторым образом для любого иного участника сети.

Аппаратные адреса в технологии Ethernet носят названия «MAC-адреса».

MAC-адрес (от англ. Media Access Control – управление доступом к носителю) – это уникальный идентификатор, сопоставляемый с различными типами оборудования для компьютерных сетей. Большинство сетевых протоколов канального уровня используют одно из трёх пространств MAC-адресов, управляемых IEEE: MAC-48, EUI-48 и EUI-64. Адреса в каждом из пространств теоретически должны быть глобально уникальными. Не все протоколы используют MAC-адреса, и не все протоколы, использующие MAC-адреса, нуждаются в подобной уникальности этих адресов.

В широковебательных сетях (таких, как сети на основе Ethernet) MAC-адрес позволяет уникально идентифицировать каждый узел сети и доставлять данные только этому узлу. Таким образом, MAC-адреса формируют основу сетей на канальном уровне, которую используют протоколы более высокого (сетевого) уровня. Для преобразования MAC-адресов в адреса сетевого уровня и обратно применяются специальные протоколы (например, ARP и RARP в сетях TCP/IP).

Адреса типа MAC-48 наиболее распространены; они используются в таких технологиях, как Ethernet, Token ring, FDDI и др. Они состоят из 48 бит.

Визуально MAC-адреса обычно записываются в виде 6 шестнадцатеричных чисел. Например, **00-06-29-C9-57-1C**.

Примечание. Для корректной работы устройств в рамках одной локальной сети **MAC-адреса устройств должны быть уникальны**. В виду того, что в учебных аудиториях предлагается использовать виртуальные машины с одного образа, после запуска машин аппаратные адреса виртуальных машин будет совпадать. Для решения этой проблемы необходимо завершить работу запущенной машины, открыть в текстовом редакторе (хорошо подходит notepad, т.к. поддерживает кодировку unicode) конфигурационный файл виртуальной машины *.vms и изменить строку, задающую аппаратный адрес, например поменяв последнее число на номер компьютера в аудитории (см. рис. 12).

```
<ethernet_adapter>
  <controller_count type="integer">1</controller_count>
  <ethernet_controller id="0">
    <virtual_network>
      <id type="bytes">76544E39925111DA821390A33B23363A</id>
      <name type="string">IBM 10/100 EtherJet PCI Management Adapter</name>
    </virtual_network>
    <ethernet_card_address type="bytes">0003FFC85703</ethernet_card_address>
  </ethernet_controller>
</ethernet_adapter>
```

Рис. 12. Задание аппаратного адреса сетевого адаптера виртуальной машины

2.5 Стек протоколов TCP/IP. IP-адресация

На программном уровне взаимодействие между участниками сети осуществляется через так называемые стеки протоколов. Самым распространенным стеком протоколов является TCP/IP, который используется как в локальных сетях, так и сети Internet.

Адресация машин, использующих стек TCP/IP осуществляется на базе IP-адресов.

IP-адрес (ай-пи адрес, Internet Protocol Address) — уникальный идентификатор (адрес) устройства, подключённого к локальной сети или интернету.

IP-адрес представляет собой 32-битовое (по версии IPv4) или 128-битовое (по версии IPv6) двоичное число. Удобной формой записи IP-адреса (IPv4) является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками, например, 192.168.0.1. (или 128.10.2.30 — традиционная десятичная форма представления адреса, а 10000000 00001010 00000010 00011110 — двоичная форма представления этого же адреса).

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень протокола IP передаёт пакеты между сетями. IP-адрес назначается либо статически (администратором), либо динамически (управляется некоторым выделенным сервером; чаще всего по протоколу DHCP) во время конфигурирования сетевых настроек.

IP-адрес состоит из двух частей: номера сети и номера узла. В случае изолированной сети её адрес может быть выбран администратором из специально зарезервированных для таких сетей блоков адресов (192.168.0.0/16, 172.16.0.0/12 или 10.0.0.0/8).

Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Internet Network Information Center, InterNIC), если сеть должна работать как составная часть Internet. Обычно поставщики услуг Internet получают диапазоны адресов у подразделений InterNIC, а затем распределяют их между своими абонентами.

Рассмотрим более подробно то, как интерпретируются IP-адреса.

IP-адреса: классы, подсети и узлы

Некоторое время назад Интернет было разделено на три основных класса сетей: класс А, класс В и класс С (классовая адресация). Сети класса А имели то свойство, что первый октет (восемь битов) IP-адреса определял собственно сеть, а оставшиеся биты использовались организацией, которая управляла сетью, для того, чтобы различать узлы сети. Большинство организаций, управляющих сетями класса А, разделяли их на подсети, добавляя к схеме адресации еще один уровень иерархии. В сетях класса В два первых октета использовались для определения сети, а оставшиеся два - для определения отдельных узлов, а в сетях класса С для определения сети отводилось три октета и лишь один для определения узлов (см. рис. 13).



Рис. 13. Классы IP-адресов

К сожалению, эта система мелких, средних и крупных сетей не всегда была удобна. Многие организации были достаточно велики, чтобы выйти за пределы сети класса С, которая могла содержать максимум 254 узла, но недостаточно велики, чтобы занять целый класс В, сеть которого могла вместить 65534 узла. Но многие из этих организаций получили все-таки сети класса В в свое распоряжение. Как следствие, свободные сети класса В были занесены в красную книгу.

Для решения этой проблемы и создания сетей, которые имели бы соответствующий требованиям размер, была разработана бесклассовая междоменная маршрутизация (Classless Inter-Domain Routing, или CIDR (произносится как «сайдр»). Как видно из названия, CIDR извлекается от классов А, В и С. В системе CIDR для идентификации сети может использоваться не фиксированное число октетов (один, два или три), но любое число битов IP адреса. Так, к примеру, если организации нужно адресное пространство примерно в четыре раза большее, чем адресное пространство сети класса В, власть предрержающие могут определить длину идентификатора сети в 14 битов, таким образом, оставляя 18 битов (в четыре раза больше узлов, чем в сети класса В) на используемое адресное пространство.

Совершенно естественно, что пришествие CIDR сделало «классовую» терминологию устаревшей, хотя она до сих пор довольно часто используется в разговорах. Итак, чтобы обозначить конкретную CIDR-сеть, следует указать конкретное значение старших битов, присваиваемое организации в записи через точку, а также число битов, определяющих сеть. Две части записи разделяются символом «слеш». 15/8 – прежняя сеть класса А, которая «начинается» с восьмибитной последовательности 00001111. Прежняя сеть класса В 128.32.0.0 теперь идентифицируется как 128.32/16. А сеть 192.168.0.128/25 состоит из 128 IP-адресов, начиная с адреса 192.168.0.128 и заканчивая адресом 192.168.0.255.

Маска подсети может задаваться аналогично записи IP-адреса. Например, запись 192.168.128.1/24 и эквивалентна маске подсети 255.255.255.0.

Формально выделение сети и номера узла на основе ip-адреса и маски сети осуществляется как:

NET = IP & MASK,

$HOST = IP \& (\sim MASK)$, где все компоненты 32-х битные числа, «&» – операция логического умножения, а «~» – инверсия бит.

Особые ip-адреса. К особым адресам относят:

- 127.0.0.0/8 – для компьютера обозначает «самого себя»
- Адреса, в которых номер узла обозначен числом, в битовой записи которого установлены все биты. Например, 192.168.192.255/24
- Некоторые другие.

2.6 Настройка сетевого стека TCP/IP для ОС семейства Windows 2000+

Настройка сетевых параметров в ОС Windows 2000 и более поздних осуществляется через свойства соответствующего сетевого адаптера (см. рис 14).

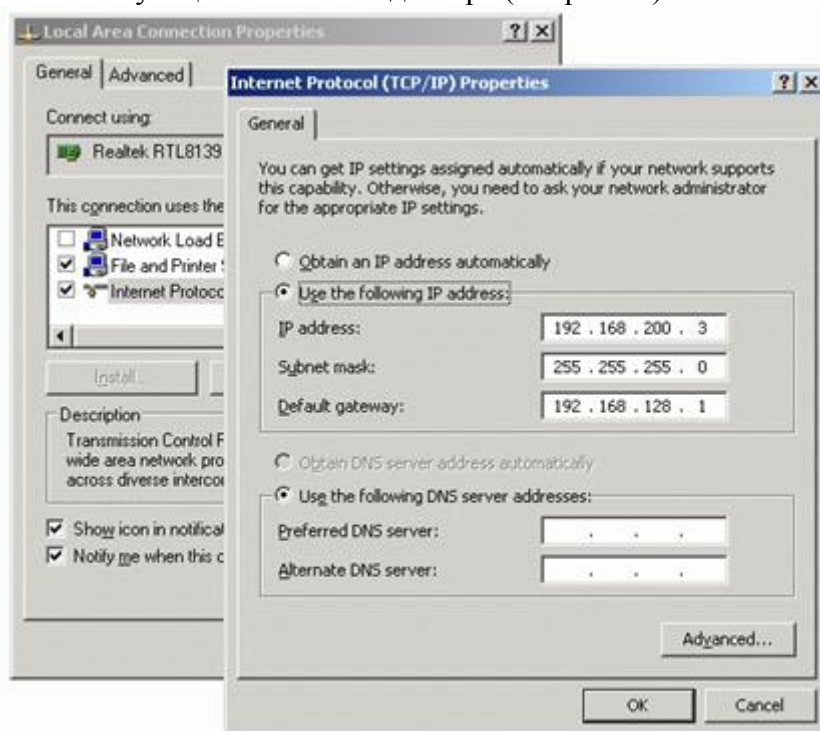


Рис. 14. Окно настройки основных параметров

В примере показано задание статических параметров. Для получения настроек динамически необходимо выбрать опции («Obtain an IP address automatically»)

Примечание. Также конфигурирование может быть осуществлено с использованием консольной утилиты **netsh** (например, см. «netsh interface /?»).

2.7 Сетевые консольные утилиты

Для работы с сетевыми настройками в консольном режиме (cmd.exe) служат несколько утилит.

Ipconfig – эта утилита инструментов для просмотра настроек сетевых соединений и устранения неисправностей TCP/IP. Она выводит информацию о каждом сетевом адаптере, в том числе о назначенном ему IP-адресе, маске подсети, адресе шлюза, MAC-адресе, адресе DNS-сервера и т. д. Чтобы получить основные сведения о сетевых устройствах, наберите в командной строке ipconfig.

Параметры утилиты ipconfig

Табл. 1

Ключи	Функции
/all	выводит полную информацию о настройках TCP/IP
/displaydns	выводит информацию из кэша DNS
/flushdns	очищает кэш DNS
/registerdns	обновляет все DHCP-адреса и регистрирует заново доменные имена
/release "<адаптер>"	освободить IP-адрес для указанного устройства
/renew "<адаптер>"	возобновляет адрес указанного устройства
/setclassid "<адаптер>"<новый код>	устанавливает код класса DHCP указанного адаптера
/showclassid "<адаптер>"	выводит код класса DHCP указанного адаптера
/?	отображение справки в командной строке

Пример использования:

```
>ipconfig /all
```

Windows IP Configuration

```
Host Name . . . . . : stuff-ctam
Primary Dns Suffix . . . . . : ctam.tu-bryansk.ru
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : ctam.tu-bryansk.ru
```

Ethernet adapter Local Area Connection 1:

```
Connection-specific DNS Suffix . : ctam.tu-bryansk.ru
Description . . . . . : IBM 10/100 EtherJet PCI Management Adapter
Physical Address. . . . . : 00-06-29-C9-57-1C
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . : Yes
IP Address. . . . . : 192.168.128.111
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.128.1
DHCP Server . . . . . : 192.168.128.1
DNS Servers . . . . . : 192.168.128.5
                        192.168.128.1
Primary WINS Server . . . . . : 192.168.128.1
Secondary WINS Server . . . . . : 192.168.128.1
```

Ping – эта команда, которая базируется на IP- и ICMP-протоколах пересылки дейтограмм и служит для проверки работоспособности каналов и узлов. Для решения поставленной задачи PING использует отклики протокола ICMP (смотри также статьи о протоколах IP и ICMP).

Применяется PING и при отладке сетевых каналов. Ниже приведены параметры утилиты и пример использования команды Ping.

Параметры утилиты ping

Табл. 2

Ключи	Функции
-t	Отправка пакетов на указанный узел до команды прерывания
-a	Определение адресов по именам узлов
-n	Число отправляемых запросов
-l	Размер буфера отправки
-f	Установка флага, запрещающего фрагментацию пакета
-i TTL	Задание времени жизни пакета (поле "Time To Live")
-v TOS	Задание типа службы (поле "Type Of Service")
-r	Запись маршрута для указанного числа переходов
-s	Штамп времени для указанного числа переходов
-j список узлов	Свободный выбор маршрута по списку узлов
-k список узлов	Жесткий выбор маршрута по списку узлов
-w интервал	Интервал ожидания каждого ответа в миллисекундах

Пример использования:

```
>ping iipo.tu-bryansk.ru
```

Pinging iipo.tu-bryansk.ru [82.179.88.34] with 32 bytes of data:

Reply from 82.179.88.34: bytes=32 time=6ms TTL=58

Reply from 82.179.88.34: bytes=32 time=5ms TTL=58

Reply from 82.179.88.34: bytes=32 time=5ms TTL=58

Reply from 82.179.88.34: bytes=32 time=7ms TTL=58

Ping statistics for 82.179.88.34:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 5ms, Maximum = 7ms, Average = 5ms

Arp - эта утилита предназначена для просмотра таблицы соответствия IP-адресов сетевых машин аппаратным (MAC) адресам их сетевых адаптеров. Она также позволяет изменять эту таблицу.

Параметры утилиты **arp**:

Табл. 3

Ключи	Функции
-a	Отображает локальную таблицу соответствия IP-адресов MAC-адресам. Если указан IP-адрес, то выводится информация из таблицы только для соответствующего компьютера. Если в системе установлено более одного сетевого адаптера, то выводится информация из таблицы ARP для всех сетевых адаптеров
-g	То же, что и -a
ip_адрес	IP-адрес
-N адрес_интерфейса	Указывает, что выводятся данные из таблицы ARP только указанного адаптера
-d	Удаляет указанный хост из таблицы ARP. При задании IP-адреса допустимо использование символа * для удаления не-

	скольких адресов. Если адрес интерфейса не указан, то соответствующие записи будут удалены из таблиц всех интерфейсов
-s	Добавляет в таблицу ARP статическую запись. Если не указан адрес интерфейса, то запись будет добавлена в таблицы всех интерфейсов. Статические записи сохраняются только на время работы компьютера - после перезагрузки статические записи требуют повторного добавления
mac_адрес	MAC-адрес. Указывается в виде 6 шестнадцатеричных чисел, разделенных дефисами

Пример использования:

>arp -a

Interface: 192.168.50.111 --- 0x10003

Internet Address	Physical Address	Type
192.168.50.1	00-02-b3-50-8d-e7	dynamic
192.168.50.6	00-02-b3-e9-aa-c5	dynamic
192.168.50.11	00-14-85-31-f4-9f	dynamic
192.168.50.18	00-06-29-89-8a-66	dynamic
192.168.50.54	00-c0-f0-56-ae-cf	dynamic
192.168.50.68	00-06-29-89-73-5d	dynamic

3. Задания для выполнения

- Необходимо установить ОС Windows (XP/2003/7 или Linux:; Ubuntu, Fedora, Knoppix, Debian, Alt linux...) на виртуальную машину Virtual PC или Oracle VM VirtualBox.
- Нстроить стек протоколов TCP/IP:
 - Установить взаимодействие между несколькими запущенными в локальной сети виртуальными машинами с использованием статически заданных IP адресов. При этом рекомендуется использовать диапазон локальных адресов 192.168.0.XX и соответствующую маску подсети 255.255.255.0.
 - Ознакомиться с принципами динамического назначения адресов:
 - при наличии DHCP сервера (адреса локальной сети лаборатории, например, подсеть 192.168.192.0, маска 255.255.255.0).
 - в отсутствии DHCP сервера (автоматически назначаемые адреса с 169.254.0.1 по 169.254.255.254, маска 255.255.0.0). В данном случае DHCP-сервер должен быть недоступен.
- Далее попытаться получить доступ к «основной сети» в лаборатории(ях). При этом реализовав как динамическое, так и статическое назначение адресов машинам.
- Протестировать рассмотренные сетевые утилиты

Дополнительное задание:

- Попробовать установить для нескольких виртуальных машин одинаковый MAC адрес (в конфигурационном файле образа для Virtual PC – *.vmc).
- Задать для двух машин одинаковый IP адрес (использовать статическое назначение адресов).

Проанализировать полученные результаты.

Примечание: приветствуется выполнение работы группами при условии работы с несколькими виртуальными машинами.

4. Варианты индивидуальных заданий.

Смотрите п. 3.

5. Контрольные вопросы.

1. Каково назначение виртуальных машин?
2. Что такое топология «звезда»?
3. Что такое коммутатор (switch)? Чем отличается от классического концентратора (hub)?
4. Что такое и где применяется MAC-адрес? Почему в рамках одной локальной сети MAC-адреса должны быть уникальны?
5. Что такое IP-адрес? Что такое маска подсети?

6. Список рекомендуемой литературы.

1. В.Г. Олифер, Н.А. Олифер, “Компьютерные сети”.
(ftp://iipo.tu-bryansk.ru/pub/Drozдов/Net/Books/book_comp_net_olifer.chm)
2. Компьютерные сети. 4-е изд. / Э. Таненбаум. – СПб.: Питер, 2006. – 992 с.: ил. – (Серия “Классика computer science”).
(<ftp://iipo.tu-bryansk.ru/pub/Drozдов/Net/Books/TanenbaumKomputernyeSeti.djvu>)

ЛАБОРАТОРНАЯ РАБОТА №3 ВЗАИМОДЕЙСТВИЕ ПРИКЛАДНЫХ ПРОГРАММ С ПОМОЩЬЮ ТРАНСПОРТНЫХ ПРОТОКОЛОВ СЕТИ ИНТЕРНЕТ

1. Цель работы

Изучение принципов работы транспортных протоколов Интернет, разработка прикладных программ, осуществляющих взаимодействие между собой на основе этих протоколов.

Для выполнения лабораторной работы требуется написать программу, которая выполняется под управлением ОС типа Windows или UNIX и использует для взаимодействия с другими программами заданный протокол сети Интернет. Для разработки программы рекомендуется использовать среду Delphi версии 3.0 или старше под управлением ОС типа Windows 95/98 или Windows NT/2000.

2. Общие сведения

2.1 Транспортный протокол TCP сети Internet

TCP (Transmission Control Protocol) - это один из самых широко распространенных протоколов транспортного уровня. Главная функция TCP заключается в доставке сообщений без потерь, чего не может гарантировать протокол более низкого уровня IP (Internet Protocol). Для доставки сообщений предварительно устанавливается соединение между процессом-отправителем и процессом-получателем. Данное соединение осуществляет надежную доставку дейтаграмм. Протокол TCP производит повторную передачу искаженного или утерянного пакета.

Выделение всех функций, необходимых для надежной доставки сообщений, в отдельный уровень освобождает разработчиков прикладных программ и утилит от решения задач управления потоком дейтаграмм. Протокол обеспечивает сквозную передачу данных от отправителя к получателю. Поскольку TCP ориентирован на установление соединения, то адресат, получивший дейтаграмму, должен уведомить отправителя об этом. Подразумевается, что между отправителем и получателем устанавливается виртуальный канал, где они обмениваются сообщениями, часть из которых есть подтверждения о получении данных либо коды ошибок. Виртуальный канал на самом деле может подразумевать несколько реальных физических каналов передачи данных, поскольку сообщение может проходить через один или несколько шлюзов.

Когда некоторое приложение (процесс) прикладного уровня отправляет сообщение другому приложению с помощью TCP, предполагается, что сообщение является потоком, т.е. представляет собой поток байтов, передаваемых асинхронно. TCP получает поток байтов и собирает его в пакеты (сегменты), добавляя заголовки в начало сегментов. Длина сегмента обычно определяется протоколом или выбирается администратором системы.

Процесс обмена данными начинается с передачи запроса на установление соединения от машины-отправителя к машине-получателю. В запросе содержится специальное целое число, называемое номером сокета (socket). В ответ получатель посылает номер своего сокета. Номера сокетов отправителя и получателя однозначно определяют соединение (конечно, соединение также не возможно без указания IP-адресов отправителя и получателя, но эта задача решается протоколами более низкого уровня - IP).

После установления соединения TCP начинает передавать сегменты сообщения. На более низком IP-уровне отправителя сегменты разбиваются на одну или несколько дейтаграмм. Пройдя через сеть, дейтаграммы поступают к получателю, где IP-уровень снова собирает из них сегменты и передает их TCP. TCP собирает все сегменты в сообщение. От TCP сообщение поступает к процессу-получателю, где обрабатывается протоколом прикладного уровня.

TCP на машине-получателе собирает целое сообщение из сегментов, руководствуясь порядковыми номерами сегментов, которые записаны в их заголовке. Если какой-то сегмент сообщения потерян или поврежден (что проверяется по контрольной сумме в заголовке сегмента), то отправителю посылается сообщение, содержащее номер ошибочного сегмента. В этом случае отправитель повторно передает сегмент. Если сегмент успешно принят, то получатель посылает отправителю подтверждение-квитанцию (АСК - acknowledgement).

В TCP применяется средство ограничения потока данных, называемое скользящим окном. Оно представляет собой фрагмент сообщения, которые адресат готов принять. При установлении соединения отправителю сообщается размер окна (размер окна кратен размеру сегмента). После того, как отправитель передал количество байтов, соответствующее размеру окна, он должен ждать квитанции. Как только будет получена квитанция на переданные сегменты, окно сдвигается вправо на соответствующее число байтов, и новые сегменты могут быть переданы. Отправитель может передать без получения квитанций в сеть максимально столько сегментов, сколько их укладывается в скользящем окне. В процессе обмена данными получатель может присылать квитанции, в которых будет указан новый размер скользящего окна.

Важную роль в протоколе TCP играют таймеры. Сегмент считается потерянным, если квитанция на него не поступила в течение заданного времени ожидания. При этом производится повторная передача сегмента. При получении квитанции таймер останавливается. Если получатель обнаруживает несколько правильных копий одного и того же сегмента, то все лишние копии просто отбрасываются и отправителю передается только одна квитанция.

2.2 Номера портов и сокет

Приложение (процесс), использующее TCP однозначно определяется числом - номером порта (сокета). В принципе, номера портов можно назначать процессам произвольно, но для облегчения взаимодействия между различными программами прикладного уровня приняты соглашения о номерах портов, закрепленных за определенными службами Internet. Номера портов наиболее известных служб сети приведены в табл. 4.

Таблица 4 Номера портов наиболее употребительных служб сети Internet

Номер порта	Служба сети	Описание
0		Зарезервирован
7	Echo	Эхо-ответ на входящие сообщения
9	Discard	Сброс (поглощение) всех входящих сообщений
11	Users	Активные пользователи
13	Daytime	Отклик, содержащий время дня
19	Chargen	Генератор символов
20	ftp data	Передача данных по протоколу FTP
21	ftp	Передача управляющих команд по протоколу FTP
23	telnet	Порт подключения по протоколу TELNET
25	Smtп	Протокол передачи почтовых сообщений SMTP
37	Time	Отклик, содержащий время
42	Name	Сервер имен
43	Whois	Кто это
53	Domain	Сервер имен доменов
67	Boots	Протокол удаленной загрузки сервера
68	Bootc	Протокол удаленной загрузки клиента

69	Tftp	Упрощенный протокол передачи файлов TFTP
79	Finger	Протокол получения информации о пользователях FINGER
80	http	Протокол передачи гипертекста HTTP
109	Pop2	Протокол почтового ящика POP2
110	Pop3	Протокол почтового ящика POP3
111	Rpc	Протокол удаленного вызова процедур RPC
156	Sqlserv	Служба SQL
161	snmp	Управляющий протокол SNMP

В формате сообщения протокола TCP под номер порта отводится 16 бит, поэтому максимально возможным номером порта является число 65535. Номера портов от 0 до 255 строго зарезервированы под системные нужды, их не допускается использовать в прикладных программах. В интервале от 256 до 1023 многие порты также используются сетевыми службами, поэтому и их не рекомендуется применять для прикладных нужд. Как правило, большинство прикладных приложений, построенных на основе TCP/IP используют номера портов в диапазоне от 1024 до 5000. Рекомендуется использовать номера от 3000 до 5000, номера выше 5000 используются чаще всего для краткосрочного применения.

Любой канал связи в TCP определяется двумя числами - эта комбинация называется сокетом. Таким образом, сокет определяется IP-адресом ЭВМ и номером порта, используемым программным обеспечением TCP. При соединении любая машина однозначно определена IP-адресом, а каждый процесс - портом, поэтому соединение между двумя процессами однозначно определяется сокетом. Схема установления соединения по протоколу TCP между тремя ЭВМ в сети отражена на рис. 6.

Взаимодействующие ЭВМ ведут таблицы всех активных портов отправителей и получателей. Если между двумя машинами происходит обмен данными, то порт, который в одной из них является отправителем, в другой будет получателем и наоборот. Если машина отправитель запрашивает несколько соединений, то у каждого из них свой порт-отправитель, а порт получателя может быть общим. Несколько машин могут одновременно использовать один и тот же порт получатель, это называется мультиплексированием. На рис. 6 мультиплексирование двух соединений выполняется на ЭВМ 3 по порту номер 23.

Когда устанавливается несколько соединений, то может случиться, что несколько машин пошлют запросы на соединение, в которых указаны одинаковые порты источники и получатели. Однако путаницы с соединениями не возникает, потому что IP-адреса у всех машин разные, следовательно, каждое соединение будет однозначно определено своим сокетом.

2.3 Программирование обмена данными на основе транспортных протоколов

TCP должен взаимодействовать не только с протоколами нижележащего уровня, но и с протоколами и приложениями прикладного уровня. Связь с прикладным уровнем осуществляется с помощью набора сервисных примитивов. Сервисные примитивы определены в стандарте протокола, а для прикладных программ они доступны в форме библиотек работы с сокетами.

При установлении соединения каждая из сторон выполняет некоторые операции, называемые открытием соединения. Открытие может быть пассивным или активным. Как правило, одна из сторон производит активное открытие соединения, а другая - пассивное, тогда соединение устанавливается. Оба режима подчиняются четким правилам. Пассивное соединение еще иногда называют серверным, а активное - клиентским.

При активном соединении процесс прикладного уровня передает программному обеспечению TCP на той же ЭВМ сервисный примитив запроса на установление соединения с номером сокета, после чего TCP отправляет получателю запрос на установление соединения,

затем ждет ответа. После установления соединения активный процесс (клиент) может инициировать прием или передачу данных.

При пассивном соединении прикладная программа переводит программное обеспечение ТСП в режим ожидания запроса на соединение от удаленной системы. Когда поступает запрос, программное обеспечение ТСП осуществляет установку соединения, после чего пассивный процесс (сервер) готов принимать и передавать данные.

Программный интерфейс сокетов был разработан для ОС UNIX. Библиотека функций, поддерживающих этот интерфейс, входит в ядро всех ОС типа UNIX и Linux. Однако принципы работы с этим программным интерфейсом применимы к большинству ОС, поддерживающих ТСП/IP (например, в семействе ОС и оболочек типа Windows программный интерфейс сокетов реализован в динамической библиотеке Winsock.dll).

Для протокола ТСП пассивное (на стороне сервера) соединение с сокетом приводит к выполнению следующих функций:

- создание сокета и установление его типа (в ОС типа UNIX функция socket);
- настройка сокета на конкретное соединение (указывает адрес и номер порта - в ОС типа UNIX - функция bind);
- создание очередь клиентов (в ОС типа UNIX - функция listen);
- ожидание приходящего запроса на соединение с сокетом (в ОС типа UNIX - функция accept);
- прием и передача данных от клиента (в ОС типа UNIX - функции read, write, send, recv и их модификации);
- закрытие соединения с клиентом (в ОС типа UNIX - функция close).

Получив входящий запрос на соединение, сервер должен решать как бы две задачи одновременно: обслуживать уже установленное с клиентом соединение в соответствии с прикладным протоколом (принимать и отдавать данные клиенту) и ожидать поступления новых запросов на соединение от других клиентов. Обычно в развитых ОС (а сюда подпадают все современные ОС за исключением, разве что, MS DOS) эта проблема решается за счет возможностей параллельного выполнения нескольких процессов. Сервер может породить новый процесс (или новую цепочку выполнения - thread), который и должен будет заняться обслуживанием уже установленного соединения, а сам основной процесс сервера может закрыть текущее соединение и вновь вернуться к ожиданию запросов на соединение от других клиентов. В ОС типа UNIX создание нового процесса решается с помощью функции fork, при этом за вновь созданным процессом сохраняются все соединения, сделанные в основном процессе.

Для протокола ТСП активное (на стороне клиента) соединение с сокетом приводит к выполнению следующих функций:

- создание сокета и установление его типа (в ОС типа UNIX функция socket);
- установление соединения с сервером (указывает адрес и номер порта - в ОС типа UNIX - функция connect);
- прием и передача данных (в ОС типа UNIX - функции read, write, send, recv и их модификации);
- закрытие соединения с сервером (в ОС типа UNIX - функция close).

Клиент, как правило, не требует для своей работы параллельного выполнения нескольких процессов. О синхронных и асинхронных сокетах можно почитать в Интернете.

В среде программирования Borland Delphi существуют специальные классы, которые позволяют выполнять все те же действия, что и библиотека сокетов в ОС UNIX. Они взаимодействуют с библиотекой Winsock.dll на основе специальных технологий ОС (ActiveX технологии и COM-объекты). В среде Borland Delphi версии 3.0 для целей клиентского и серверного соединений служит класс объектов TTCP; а в среде Borland Delphi версии 5.0 для клиентского

соединения существует класс объектов TClientSocket, а для серверного - TServerSocket. Естественно, пользователь может на основе базовых классов разрабатывать свои собственные классы, которые будут поддерживать соединения по определенным им самим прикладным протоколам.

Для того чтобы создать сокет, достаточно создать экземпляр объекта выбранного класса (TTCPS - в среде Borland Delphi версии 3.0 на страничке компонент "Internet", TClientSocket или TServerSocket - в среде Borland Delphi версии 5.0 также на страничке компонент "Internet"). Это можно выполнить при проектировании приложения в среде разработки или же средствами языка программирования при выполнении приложения. Чтобы специфицировать (настроить) сокет необходимо созданному экземпляру объекта присвоить нужные значения в указанные свойства (properties) - как правило, это свойства с именами вида "Port" и "Host" (имена и состав свойств зависят от версии среды разработки). Это тоже можно сделать как в режиме проектирования приложения, так и командами присвоения свойств объекта в тексте программы. После этого сокет инициализирован и с ним можно работать.

Для работы сокета клиента необходимо открыть сокет (процедура Open), затем использовать процедуры установления соединения, передачи и приема данных, а в конце работы закрыть сокет (процедура Close). При удалении экземпляра объекта автоматически прекратит существование и связанный с ним сокет.

Для работы сокета сервера после его создания надо вызвать процедуру ожидания соединения. При этом по установлению соединения наступит событие, которое программист должен соответствующим образом обработать. В среде Borland Delphi версии 3.0 программист сам должен создавать потоки выполнения для обслуживания соединения (для выполнения потоков служат экземпляры класса объектов TThread), а в среде Borland Delphi версии 5.0 это можно сделать в автоматическом или полуавтоматическом режиме. После окончательного завершения работы сокет сервера следует закрыть. При удалении экземпляра объекта автоматически прекратит существование и связанный с ним сокет.

Более подробная информация о функциях программного интерфейса с описанными компонентами фирмы Borland и достаточная для выполнения лабораторной работы написана в двух прилагаемых статьях (или можете найти сами в специальной литературе и Internet).

1- Программирование сокетов

С помощью следующего приложения мы осветим вопросы программирования сокетов как для протокола TCP, так и для протокола UDP. Приложение функционирует следующим образом.

1. Клиент считывает со стандартного устройства ввода (клавиатуры) строку символов и посылает эту строку серверу через свой сокет.
2. Сервер принимает строку через свой сокет.
3. Сервер переводит все символы строки в верхний регистр.
4. Сервер отправляет модифицированную строку клиенту.
5. Клиент получает строку и печатает ее с помощью стандартного устройства вывода (монитора).

Мы начнем с рассмотрения приложения, в котором взаимодействие клиента и сервера осуществляется через логическое соединение по протоколу TCP (рис. 2.20).

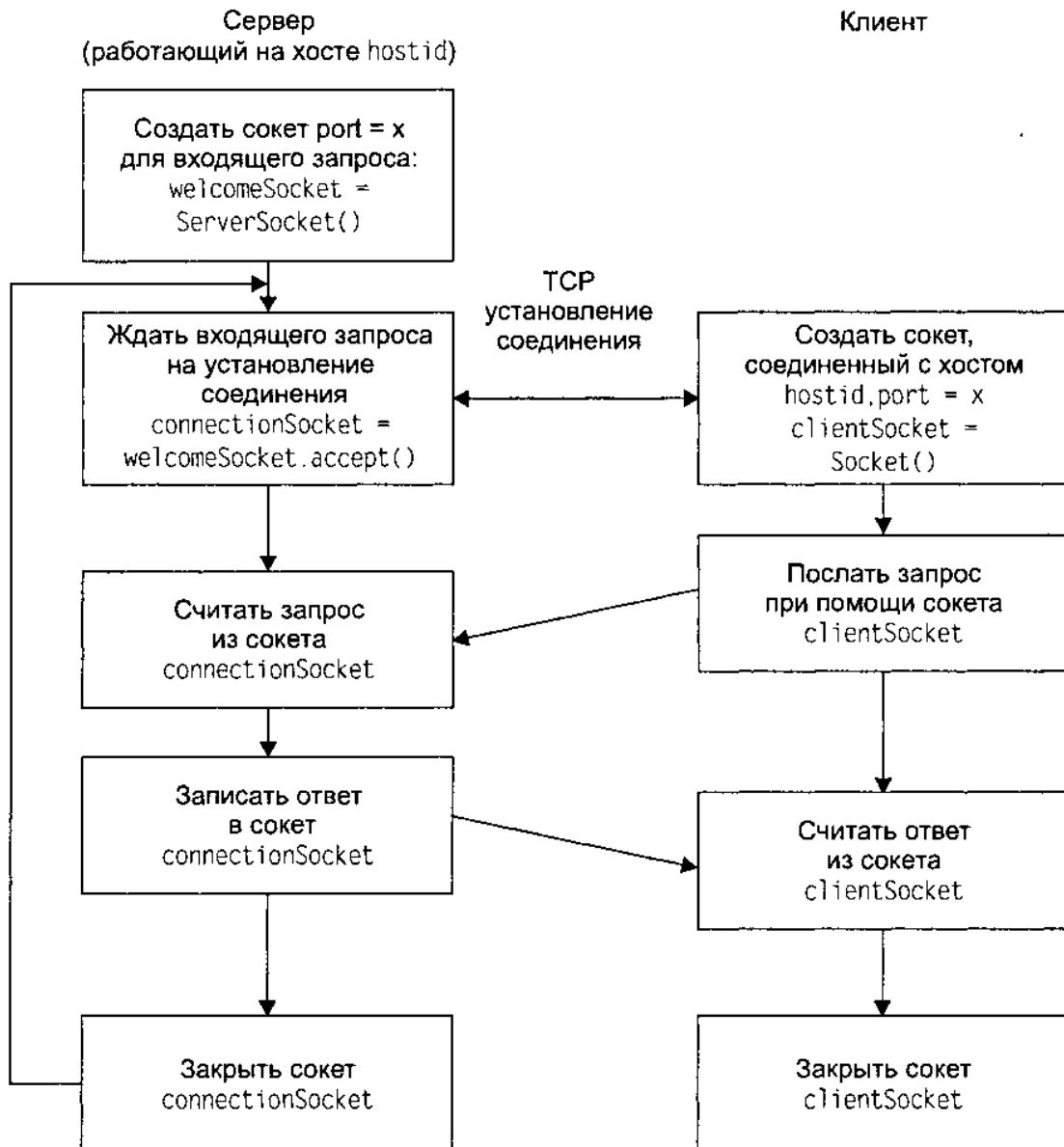


Рис. 2.20. Приложение клиент/сервер, использующее транспортные службы с установлением логического соединения

Ниже мы приведем тексты программ для клиентской и серверной сторон приложения и снабдим пояснениями каждую строку кода. Программа-клиент названа `TCPClient.java`, программа-сервер — `TCPServer.java`. Отметим, что приведенные программы лишь иллюстрируют ключевые фрагменты приложения и не претендуют на то, чтобы называться «хорошими» с точки зрения практического применения. Для улучшения программ приведенные тексты следовало бы снабдить несколькими дополнительными строками.

После компиляции обеих программ (каждой на своем компьютере) первой запускается программа-сервер, поскольку для установления соединения серверный процесс должен ожидать запроса со стороны клиентского процесса. Клиентский процесс создается после запуска программы-клиента, и в его функцию входит инициирование TCP-соединения с сервером. После того как соединение установлено, пользователь, находящийся на клиентской стороне, может вводить строки символов и получать их обратно в верхнем регистре.

Ниже приведен текст программы `TCPClient.java`.

```
import java.io.*;
import java.net.*;
```



```
class TCPClient {
public static void main(String argv[]) throws Exception
{
String sentence;
String modifiedSentence;
BufferedReader inFromUser =
new BufferedReader(
new InputStreamReader(System.in));
Socket clientSocket = new Socket ("hostname". 6789);
DataOutputStream outToServer =
new DataOutputStream(
clientSocket.getOutputStream());
BufferedReader inFromServer =
new BufferedReader(
new InputStreamReader(
cli entSocket.getInputStream()));
sentence = inFromUser.readLine();
outToServer.writeBytes(sentence + '\n');
modifiedSentence = inFromServer.readLine();
System.out.printing "FROM SERVER: " + modifiedSentence);
clientSocket.close();
}
}
```

Как показано на рис. 2.21, программа TCPClient создает три потока данных и один сокет. Сокет имеет название clientSocket. Поток inFromUser является входным потоком данных программы и связан со стандартным устройством ввода (клавиатурой). Все символы, вводимые с клавиатуры пользователем, попадают в поток inFromUser. Другой входной поток данных программы, inFromServer, связан с сокетом и состоит из символов, передаваемых серверной стороной приложения. Выходной поток данных программы TCPClient имеет название outToServer и также связан с сокетом. Этот поток содержит символы, передающиеся серверу для обработки.

Теперь рассмотрим приведенный код подробнее. Первые две строки содержат имена двух Java-пакетов, java i'o и java. net:

```
import java.io.*;
import java.net.*;
```

Пакет java.io содержит классы входных и выходных потоков данных. Обычно этими классами являются BufferedReader и DataOutputStream, которые и были использованы в программе. Пакет java.net хранит классы, поддерживающие работу с компьютерной сетью (Socket и ServerSocket). Объект clientSocket программы порожден классом Socket.

```
class TCPClient {
public static void main(String argv[]) throws Exception
{.....}
}
```

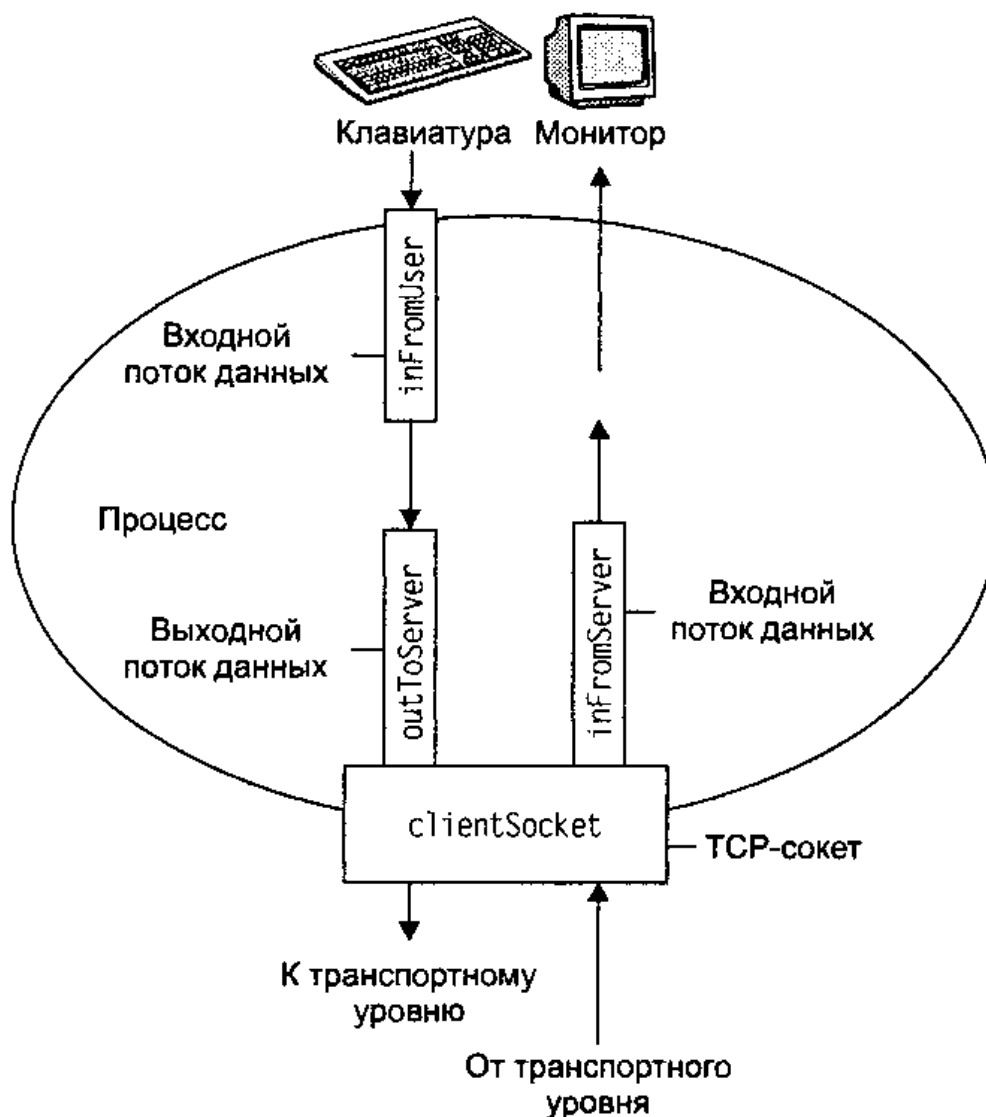


Рис. 2.21. Программа TCPClient организует три потока данных для символов

Конструкции, аналогичные приведенной выше, являются стандартными в практике программирования на Java. Первая строка представляет собой начало блока определения класса. В ней присутствует ключевое слово `class` и имя определяемого класса `TCPClient`. Класс может содержать переменные и методы; в объявлении класса они заключены в фигурные скобки и фактически составляют блок определения класса. В классе `TCPClient` нет переменных, он содержит единственный метод с именем `main()`. Методы похожи на процедуры и функции языков, подобных C; метод `main()` также играет роль, схожую с функцией `main()` в C или C++. Когда интерпретатор Java исполняет приложение (будучи вызванным управляющим классом приложения), он обращается к методу `main()` этого класса. Метод `mainQ` вызывает все остальные методы, используемые в приложении. Если в данный момент вы впервые сталкиваетесь с Java-приложением, можете не обращать внимания на ключевые слова `public`, `static`, `void`, `main` и `throws Exception`.

```
String sentence;  
String modifiedSentence;
```

Приведенные две строки являются объявлениями объектов типа `String`. Объект `sentence` предназначен для хранения строки, вводимой пользователем и передаваемой серверу. Объект

modifiedSentence содержит строку, принимаемую от сервера и выводимую на стандартное устройство вывода.

```
BufferedReader inFromUser =  
new BufferedReader(new InputStreamReader(System.in));
```

Здесь происходит создание потокового объекта in From User типа BufferedReader. Инициализация потока данных производится объектом System.in, связывающим поток со стандартным устройством ввода. После выполнения этой команды клиенту начинают передаваться символы, вводимые пользователем с клавиатуры.

```
Socket clientSocket = new Socket ("hostname", 6789);
```

В приведенной строке происходит создание объекта clientSocket типа Socket. Кроме того, при этом иницируется TCP-соединение между клиентом и сервером. Слово hostname необходимо заменить именем хоста, сохранив кавычки (например, "_fling.seas.upenn.edu"). Перед началом установки TCP-соединения клиент производит DNS-запрос IP-адреса хоста. Значение 6789 — это номер порта; вы можете выбрать другое число, однако необходимо помнить, что клиентская и серверная стороны должны использовать один и тот же номер порта. Как упоминалось ранее, IP-адрес хоста в совокупности с номером порта приложения идентифицирует серверный процесс.

```
DataOutputStream outToServer =  
new DataOutputStream(clientSocket.getOutputStream());  
BufferedReader inFromServer =  
new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Здесь происходит создание потоковых объектов, связанных с сокетом. Поток outToServer обеспечивает вывод программы через сокет, а поток inFromServer предназначен для приема данных от серверной стороны (см. рис. 2.21).

```
sentence = inFromUser. readLine();
```

Эта строка помещает последовательность символов, вводимых пользователем, в переменную sentence. Ввод оканчивается нажатием пользователем клавиши Enter. Как видим, для помещения символов в переменную используется потоковый объект inFromUser.

```
outToServer.writeBytes(sentence + '\n');
```

Здесь строка sentence, снабженная символом возврата каретки, помещается в выходной поток outToServer. После этого она будет передана через сокет в канал, соединяющий клиента с сервером.

```
modifiedSentence = inFromServer. readLine();
```

Ответ сервера принимается во входной поток inFromServer, откуда принятая строка копируется в переменную modifiedSentence. Помещение символов в строку modifiedSentence продолжается до тех пор, пока не будет получен символ возврата каретки.

```
System.out.println("FROM SERVER: " + modifiedSentence);
```

Здесь происходит вывод принятой от сервера строки на монитор,

```
clientSocket.close();
```

Эта строка закрывает сокет, а следовательно, и TCP-соединение между клиентом и сервером. Как будет показано в главе 3, выполнение этой команды ведет к отправке TCP-клиентом сообщения TCP-серверу.

Ниже приведен текст программы TCPServer.java.

```
import java.io.*;  
import java.net.*;
```

```
class TCPServer {
public static void main(String argv[]) throws Exception
{
String clientSentence;
String capitalizedSentence;
ServerSocket welcomeSocket = new ServerSocket (6789);
while (true) {
Socket connectionSocket = welcomeSocket.
accept();
BufferedReader inFromClient =
new BufferedReader(new InputStreamReaderC
connectionSocket.getInputStream()));
DataOutputStream outToClient =
New DataOutputStreamC connectionSocket.getOutputStream());
clientSentence = inFromClient. readLine(); capitalizedSentence =
clientSentence.toUpperCase() + '\n';
outToClient.writeBytes(capitalizedSentence);
}
}
}
```

Программа TCPServer имеет много общего с программой TCPClient; рассмотрим ее подробнее, опустив строки, общие с программой TCPClient и описанные выше.

```
ServerSocket welcomeSocket = new ServerSocket (6789);
```

Здесь происходит создание объекта welcomeSocket типа ServerSocket. Объект welcomeSocket представляет собой впускающий сокет, с помощью которого клиент устанавливает первоначальный контакт с сервером. Для этого сокета используется порт с номером 6789, совпадающим с номером порта сокета клиента (мы раскроем причины совпадения номеров портов сокетов клиента и сервера в главе 3). Протокол TCP устанавливает прямой виртуальный канал между сокетом clientSocket на клиентской стороне и connectionSocket на серверной стороне, после чего клиент и сервер могут свободно осуществлять обмен информацией. После создания сокета connectionSocket сервер может вновь использовать сокет welcomeSocket для инициирования других соединений с клиентами (приведенная программа не обрабатывает новых соединений, однако ее можно модифицировать соответствующим образом).

Далее программа создает несколько потоковых объектов, аналогичных clientSocket.

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

та команда основная в приложении. Она выполняет получение строки, передаваемой клиентом, перевод строки в верхний регистр с помощью метода toUpperCase() и добавление в конец символа возврата каретки. Все остальные команды программы являются периферийными и не касаются взаимодействия сервера с клиентом.

Для того чтобы протестировать совместную работу наших программ, следует поместить их на разные хосты и указать в программе TCPClient имя хоста-сервера. Затем необходимо запустить программу TCPServer, чтобы создать серверный процесс. Серверный процесс будет находиться в состоянии ожидания до тех пор, пока клиентский процесс не иницирует TCP-соединение; для этого, в свою очередь, нужно запустить программу TCPClient. Наконец,

чтобы проверить наше приложение в действии, следует ввести на стороне клиента строку, оканчивающуюся символом возврата каретки (он вводится нажатием клавиши Enter).

2- Создаем клиент-сервер на сокетах

1. Что такое сокеты?
2. Создаем сервер
3. Создаем клиент
4. Замечания

Что такое сокеты?

Для начала давайте определим что такое сервер и клиент. Итак, сервер - это специальная программа, обычно запущенная на отдельном компьютере (хосте, от слова host(eng.) - хозяин), и выполняющая некий круг задач. Клиент, в свою очередь - программа, которая запрашивает сервер выполнить то или иное действие (задачу) и вернуть полученные данные клиенту. На хосте для работы сервера обычно выделяется порт (port). К этому порту и должен будет обращаться клиент. Клиент для связи с портом хоста, который соединен в свою очередь с нужным сервером (программой), создает сокет.

В целом алгоритм работы системы клиент-сервер выглядит следующим образом:

1. Сервер подключается к порту на хосте и ждет соединения с клиентом;
2. Клиент создает сокет и пытается соединить его с портом на хосте;
3. Если создание сокета прошло успешно, то сервер переходит в режим ожидания команд от клиента;
4. Клиент формирует команду и передает ее серверу, переходит в режим ожидания ответа;
5. Сервер принимает команду, выполняет ее и пересылает ответ клиенту.
6. и т.д.

Теперь попробуем создать сервер и клиент.

Создаем сервер

Для создания сокетов и управления ими в Java есть специальные классы `java.net.Socket` и `java.net.ServerSocket`. Первый для клиента, второй для сервера. Так же нам будут необходимы два класса из пакета `java.io.*`: `BufferedReader` и `PrintWriter` для чтения/записи в сокет (думаю, что читатель уже знаком с этими классами).

Для начала подключимся к порту хоста. Сделать это можно с помощью конструктора класса `ServerSocket`. Обратите внимание, что конструктор выбрасывает исключение типа `IOException`, т.е. нам понадобится блок `try - catch`:

```
ServerSocket servers;  
try {  
    servers = new ServerSocket(4444);  
} catch (IOException e) {  
    System.out.println("Couldn't listen to port 4444");  
    System.exit(-1);  
}
```

После успешного подключения к порту сервер должен ждать подключения от клиента. Сделать это можно так:

```
Socket fromclient;  
try {  
    System.out.print("Waiting for a client...");  
    fromclient= servers.accept();  
    System.out.println("Client connected");  
} catch (IOException e) {  
    System.out.println("Can't accept");  
    System.exit(-1);  
}
```

Рассмотрим этот блок подробнее. Метод `servers.accept()` позволяет серверу следить за портом, или иначе говоря ждать подключения клиента. Как только клиент подключается - сокет для клиента сразу же создается. В противном случае выбрасывается исключение `IOException`.

Как только клиент подключился к серверу, сервер должен создать потоки ввода и вывода для связи с ним. Это можно сделать следующим образом:

```
BufferedReader in;  
PrintWriter out;  
in=new BufferedReader(new  
    InputStreamReader(fromclient.getInputStream()));  
out = new PrintWriter(fromclient.getOutputStream(),true);
```

И далее можно просто считывать данные из потока `in` и записывать данные в `out`.

Исходный код сервера:

```
import java.io.*;  
import java.net.*;  
  
public class Server {  
  
    public static void main(String[] args) throws IOException {  
        System.out.println("Welcome to Server side");  
        BufferedReader in = null;  
        PrintWriter out= null;  
  
        ServerSocket servers = null;  
        Socket fromclient = null;  
  
        // create server socket  
        try {  
            servers = new ServerSocket(4444);  
        } catch (IOException e) {  
            System.out.println("Couldn't listen to port 4444");  
            System.exit(-1);  
        }  
  
        try {  
            System.out.print("Waiting for a client...");  
            fromclient= servers.accept();  
            System.out.println("Client connected");  
        } catch (IOException e) {  
            System.out.println("Can't accept");  
        }  
    }  
}
```

```
        System.exit(-1);
    }

    in  = new BufferedReader(new
        InputStreamReader(fromclient.getInputStream()));
    out = new PrintWriter(fromclient.getOutputStream(),true);
    String      input,output;

    System.out.println("Wait for messages");
    while ((input = in.readLine()) != null) {
        if (input.equalsIgnoreCase("exit")) break;
        out.println("S ::: "+input);
        System.out.println(input);
    }
    out.close();
    in.close();
    fromclient.close();
    servers.close();
}
}
```

Создаем клиент

Для создания клиента достаточно класса Socket и двух классов для ввода/вывода (см. Создаем сервер). Также необходимо знать имя компьютера (хоста), на котором запущен сервер и номер порта.

Конструктор сокета имеет два параметра: имя хоста и номер порта. Опять таки, конструктор выбрасывает исключение типа IOException.

```
Socket fromserver = null;
fromserver = new Socket("localhost",4444);
```

Далее аналогичным образом, как для сервера, создаем потоки ввода вывода. И можно записывать/считывать данные.

Исходный код клиента:

```
import java.io.*;
import java.net.*;

public class client {
    public static void main(String[] args) throws IOException {

        System.out.println("Welcome to Client side");

        Socket fromserver = null;

        if (args.length==0) {
            System.out.println("use: client hostname");
            System.exit(-1);
        }
    }
}
```

```
System.out.println("Connecting to... "+args[0]);

fromserver = new Socket(args[0],4444);
BufferedReader in  = new
    BufferedReader(new
        InputStreamReader(fromserver.getInputStream()));
PrintWriter      out = new
    PrintWriter(fromserver.getOutputStream(),true);
BufferedReader inu = new
    BufferedReader(new InputStreamReader(System.in));

String fuser,fserver;

while ((fuser = inu.readLine())!=null) {
    out.println(fuser);
    fserver = in.readLine();
    System.out.println(fserver);
    if (fuser.equalsIgnoreCase("close")) break;
    if (fuser.equalsIgnoreCase("exit")) break;
}

out.close();
in.close();
inu.close();
fromserver.close();
}
}
```

Замечания

Клиент-сервер организован на примере Echo server (Эхо сервер). Клиент получает обратно строку переданную серверу.

Несколько замечаний по кодам:

Обратите внимание на конструкцию методов main :

```
public static void main(String[] args) throws IOException {}
```

throws IOException позволит не использовать блоки try-catch для ловки исключения IOException в самом методе, что достаточно удобно.

В клиентской части используется параметр командной строки для указания имени хоста. Например, если Вы запускаете сервер и клиент на одном компьютере, то клиент надо запускать так:

```
java client localhost
```


3- Клиент-сервер на java

Пример клиент-серверного приложения на JAVA.

Для начала нужно создать два приложения Client и Server:

```
touch Client.java
```

```
touch Server.java
```

Приложение Server будет висеть на определенном порту и слушать все обращения к нему, обрабатывать их и возвращать результат клиенту, делается это через socket-ы.

Содержимое файла Server.java:

```
import java.net.*;
import java.io.*;

class Server {
    public static void main(String[] args) {
        try{
            ServerSocket server = null;
            Socket client;

            server = new ServerSocket(1234); // слушаем порт 1234
            client = server.accept();

            System.out.println("Got client");

            PrintWriter out = null;
            BufferedReader in = null;

            in = new BufferedReader(new InputStreamReader(client.getInputStream()));
            out = new PrintWriter(client.getOutputStream(), true);

            String input, output;

            while ((input = in.readLine()) != null) {
                if (input.equalsIgnoreCase("exit")) {
                    break;
                }
            }
        }
    }
}
```

```
}

// Ответ клиенту будет в виде Server: сообщение_клиента
out.println("Server:" + input);
System.out.println(input);
}
out.close();
in.close();
client.close();
server.close();
System.out.println("Server closed");
} catch (Exception e) {
System.out.println(e.getMessage());
}

}

}
```

Приложение Client будет коннектиться к определенному серверу по ip, на определенный ранее порт.

Код файла Client.java:

```
import java.net.*;
import java.io.*;

class Client {

    public static void main(String[] args) {
        try {
            Socket clientSocket = null;

            // 127.0.0.1 - IP где запущен Server, 1234 - порт
            clientSocket = new Socket("127.0.0.1", 1234);

            BufferedReader in = new BufferedReader(new
            InputStreamReader(clientSocket.getInputStream()));

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
```

```
BufferedReader inu = new BufferedReader(new InputStreamReader(System.in));

String fuser, fserver;

out = new PrintWriter(clientSocket.getOutputStream(), true);
in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

while ((fuser = inu.readLine()) != null) {
    out.println(fuser);
    fserver = in.readLine();
    System.out.println(fserver);
    if (fuser.equalsIgnoreCase("close")) {
        break;
    }
    if (fuser.equalsIgnoreCase("exit")) {
        break;
    }
}

out.close();
in.close();
inu.close();
clientSocket.close();
} catch (Exception e) {
    System.out.println(e.getMessage());
}

}

}
```

Теперь приложения нужно скомпилировать

```
javac Server.java
javac Client.java
```

Запустить сервер

```
java Server
```

Запустить клиент

```
java Client
```

```
message test
```

```
Server: message test
```

Полученные сервером сообщения можно обработать как угодно, переслать другому (всем) клиентам, запустить в системе на выполнение, etc.

3. Задания для выполнения

1. Получить вариант задания у преподавателя.
2. Разработать прикладную программу в соответствии с заданием.
3. Написать и отладить программу на ЭВМ.
4. Сдать работающую программу преподавателю.
5. Подготовить и защитить отчет.

4. Варианты индивидуальных заданий.

1. Организовать взаимодействие типа клиент - сервер. Клиент делает запрос серверу на выполнение какой-либо команды. Сервер выполняет эту команду и возвращает результаты клиенту.
2. Организовать взаимодействие типа клиент - сервер. Клиент делает запрос серверу о передаче файлов с определенным расширением из указанной директории. Сервер сканирует указанную директорию и отправляет клиенту список файлов, удовлетворяющих запросу.
3. Организовать взаимодействие типа клиент - сервер. Сервер при подключении к нему нового клиента высылает список IP-адресов уже подключенных клиентов. А остальным клиентам рассылается сообщение в виде IP-адреса о том, что подключился такой-то клиент.
4. Организовать взаимодействие типа клиент - сервер. Клиент при входе в связь с сервером должен ввести пароль. Разрешено сделать три попытки. Если пароль не верен, сервер должен блокировать IP-адрес клиента на 5 минут.
5. Организовать взаимодействие типа клиент - сервер. Клиенты подключаются к первому серверу, и передают запрос на получение определенного файла. Если этого файла нет, сервер подключается ко второму серверу и ищет файл там. Затем либо найденный файл пересылается клиенту, либо высылается сообщение, то такого файла нет.
6. Организовать взаимодействие типа клиент - сервер. К серверу одновременно может подключиться только один клиент. Остальные клиенты заносятся в очередь, и им высылается сообщение об ожидании освобождения сервера.
7. Организовать взаимодействие типа клиент - сервер. Клиент отправляет строку серверу. Сервер отправляет данную строку на другие сервера, список которых хранится в файле, а там

уже осуществляется поиск файлов содержащих данную строку. Результаты поиска отсылаются клиенту.

8. Эмуляция DNS сервера. Клиент подсоединяется к серверу, IP которого хранится в файле `dns.url` и делает ему запрос на подключение к серверу "Имя сервера". DNS-сервер имеет список, хранящийся в файле о соответствии имен серверов и IP-адресов. Если в списке нет "имени сервера" запрошенного клиентом, то сервер DNS подключается последовательно к другим серверам, хранящимся в файле `dns.url` и т.д. Если сервер не найден, клиенту возвращается соответствующее сообщение.
9. Организовать чат. К серверу подключаются клиенты. При подключении клиента сервер спрашивает имя, под которым клиент будет известен в соединении. Сервер хранит IP-адреса подключаемых клиентов и их имена. Все сообщения каждого клиента рассылаются остальным в виде "имя клиента" - сообщение". Сообщения рассылаются сервером всем клиентам также при вхождении в связь нового клиента, и выходе какого-либо клиента.

5. Контрольные вопросы.

1. Какова структура IP-адреса?
2. Как поместить и извлечь IP-адрес из структуры сокета?
3. В чем разница между моделями TCP-соединения и дейтаграмм?
4. Каковы основные шаги межпроцессного взаимодействия в модели TCP-соединения?
5. Каковы основные шаги межпроцессного взаимодействия в модели дейтаграмм?
6. Как занести в структуру сокета IP-адрес своего компьютера?
7. Каким образом извлечь информацию о клиенте после установки TCP-соединения?
8. Какова реакция системных вызовов отправки и приема сообщений в модели TCP-соединения при разрыве связи?

ЛАБОРАТОРНАЯ РАБОТА №4 ПРОТОКОЛЫ МАРШРУТИЗАЦИИ

1. Цель работы

Изучить процесс маршрутизации в сети, его задачи, маршруты, протоколы, алгоритмы а также рассмотреть алгоритм Беллмана-Форда.

2. Общие сведения

2.1 Основы маршрутизации

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. **Маршрутизатор** - это устройство, которое собирает информацию о топологии межсетевых соединений и пересылает пакеты сетевого уровня в сеть назначения.

Чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно совершить некоторое количество транзитных передач между сетями, или **хопов** (от слова hop - прыжок), каждый раз выбирая подходящий маршрут. Таким образом, **маршрут** представляет собой последовательность маршрутизаторов, через которые проходит пакет.

- Сетевой уровень должен обеспечить доставку пакета:
 - между любыми двумя узлами сети с произвольной топологией;
 - между любыми двумя сетями в составной сети.

Сеть - совокупность компьютеров, использующих для обмена данными единую сетевую технологию.

Маршрут - последовательность прохождения пакетом маршрутизаторов в составной сети.

2.2 Задачи маршрутизации

Проблема выбора наилучшего пути называется маршрутизацией, и ее решение является одной из главных задач сетевого уровня. Эта проблема осложняется тем, что **самый короткий путь - не всегда самый лучший**. Критерием при выборе маршрута может служить *время передачи данных*. Время зависит от пропускной способности каналов связи и интенсивности трафика, которая может с течением времени изменяться. Выбор маршрута может осуществляться и по другим критериям, таким как надежность передачи. Функции сетевого уровня шире, чем функции передачи сообщений по связям с нестандартной структурой, которые мы рассмотрели на примере объединения нескольких локальных сетей. Сетевой уровень также решает задачи согласования разных технологий, упрощения адресации в крупных сетях и создания надежных и гибких барьеров на пути нежелательного трафика между сетями.

2.3 Маршруты движения пакетов

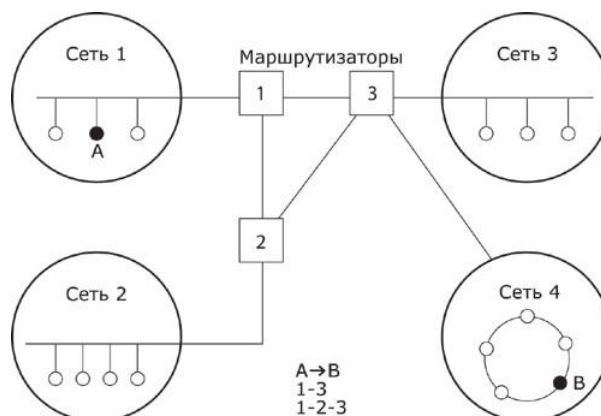


Рис. 1.

На рис. 1 показаны четыре сети, связанные тремя маршрутизаторами. Между узлами А и В данной сети пролегает два маршрута:
первый - через маршрутизаторы 1 и 3;
второй - через маршрутизаторы 1, 2 и 3.

2.4 Маршрутизация

Маршрутизатор - это устройство, распределяющее пакеты по сети с помощью информации сетевого уровня. Маршрутизатор извлекает данные об адресации сетевого уровня из пакета данных. В маршрутизаторе также имеются алгоритмы, называемые протоколами маршрутизации, с помощью которых он строит таблицы. В соответствии с этими таблицами и определяется тот маршрут, по которому должен быть направлен пакет, чтобы достичь конечного пункта назначения. Если маршрутизатор является многопротокольным, т.е. понимает несколько форматов адресов сетевого уровня и может работать с несколькими протоколами маршрутизации, к каковым и относятся маршрутизаторы компании Cisco, то он хранит отдельные таблицы маршрутизации для каждого из протоколов сетевого уровня, маршрутизация которого осуществляется.

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

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 маршрутизируемого 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).

2.5 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.

Маршрут по умолчанию - запись в таблице маршрутизации, которая используется для направления кадров, которые не имеют в таблице маршрутизации явно указанного следующего перехода. Маршруты по умолчанию могут устанавливаться как результат статического конфигурирования, выполняемого администратором.

Вообще говоря, содержание сведений обо всех других сетевых комплексах, доступных через сеть Internet, излишне и неразумно, если не невозможно. Вместо сведений о каждой конкретной сети каждому маршрутизатору компании X сообщается маршрут по умолчанию, с помощью которого он может добраться до любого неизвестного пункта назначения, направляя пакет в сеть Internet.

2.6 Адаптация к изменениям топологии

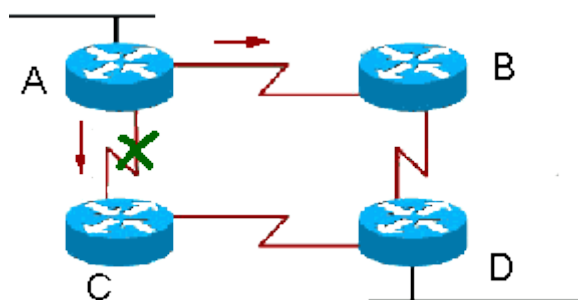


Рис. 4.6.1

Динамическая маршрутизация позволяет маршрутизаторам автоматически использовать резервные маршруты.

Сеть по-разному адаптируется к изменениям в топологии, в зависимости от того, используется статическая или динамическая информация.

Статическая маршрутизация позволяет маршрутизаторам правильно направлять пакет от сети к сети. Маршрутизатор просматривает свою таблицу маршрутизации и, следуя содержащимся там статическим данным, ретранслирует пакет маршрутизатору D (Рис. 4.6.1). Маршрутизатор D делает то же самое и ретранслирует пакет маршрутизатору C. Маршрутизатор C доставляет пакет хост-машине получателя.

Но что произойдет, если путь между маршрутизаторами A и D становится непроходимым? Ясно, что маршрутизатор A не сможет ретранслировать пакет маршрутизатору D по статическому маршруту. Связь с сетью пункта назначения будет невозможна до тех пор, пока маршрутизатор A не будет реконфигурирован на ретрансляцию пакетов маршрутизатору B.

Динамическая маршрутизация обеспечивает более гибкое и автоматическое поведение. В соответствии с таблицей маршрутизации, генерируемой маршрутизатором A, пакет может достичь своего пункта назначения по предпочтительному маршруту через маршрутизатор D. Однако к пункту назначения возможен и другой путь через маршрутизатор B. Когда маршрутизатор A узнает, что канал на маршрутизатор D нарушен, он перестраивает свою таблицу маршрутизации, делая предпочтительным путь к пункту назначения через маршрутизатор B,

а маршрутизаторы продолжают слать пакеты по этому каналу связи. Когда путь между маршрутизаторами А и D восстанавливается, маршрутизатор А может снова изменить свою таблицу маршрутизации и указать предпочтительным путь к сети пункта назначения против часовой стрелки через маршрутизаторы D и С. Протоколы динамической маршрутизации могут также перенаправлять трафик между различными путями в сети.

2.7 Операции динамической маршрутизации

Успех динамической маршрутизации зависит от двух основных функций маршрутизатора:

- Ведение таблицы маршрутизации;
- Своевременное распространение информации - в виде пакетов актуализации - среди других маршрутизаторов (рис. 11.11).

В обеспечении коллективного пользования информацией о маршрутах динамическая маршрутизация полагается на протокол маршрутизации. Протокол маршрутизации определяет набор правил, используемых маршрутизатором при его общении с соседними маршрутизаторами.

Например, протокол маршрутизации описывает следующее:

- ✓ как посылаются пакеты актуализации;
- ✓ какие сведения содержатся в таких пакетах актуализации;
- ✓ когда следует посылать эту информацию;
- ✓ как определять получателей этих пакетов актуализации;

2.8 Представление расстояния с помощью метрики

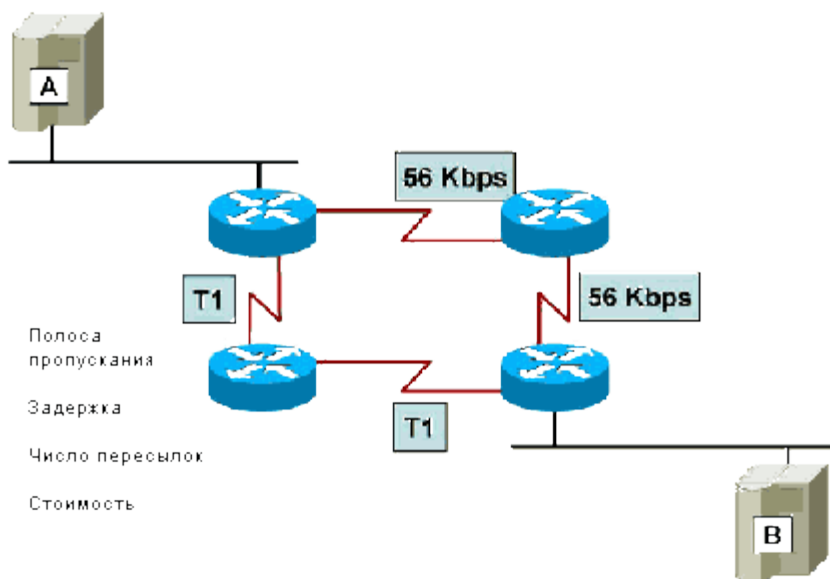


Рис. 4.6.2 Метрики маршрутизации

Когда алгоритм маршрутизации обновляет таблицу маршрутизации, его главной целью является определение наилучшей информации для включения в таблицу. Каждый алгоритм маршрутизации интерпретирует понятие «наилучшая» по-своему. Для каждого пути в сети алгоритм генерирует число, называемое метрикой. Как правило, чем меньше величина этого числа, тем лучше путь.

Метрики могут рассчитываться на основе одной характеристики пути. Объединяя несколько характеристик, можно рассчитывать и более сложные метрики. Как показано на рис. 4.6.2, при вычислении значения метрики используется несколько характеристик пути.

Наиболее общеупотребительными метриками, используемыми маршрутизаторами, являются следующие:

- Количество переходов - количество маршрутизаторов, через которые должен пройти пакет, чтобы дойти до получателя. Чем меньше количество переходов, тем лучше путь. Для обозначения суммы переходов до пункта назначения используется термин длина пути.
- Полоса пропускания - пропускная способность канала передачи данных. Например, для арендуемой линии 64 Кбит/с обычно предпочтительным является канал типа T1 с полосой пропускания 1,544 Мбит/с.
- Задержка - продолжительность времени, требующегося для перемещения пакета от отправителя получателю.
- Нагрузка - объем действий, выполняемый сетевым ресурсом, например маршрутизатором или каналом.
- Надежность - темп возникновения ошибок в каждом сетевом канале. Тик - задержка в канале передачи данных, определяемая в машинных тактах IBM-подобного ПК (приблизительно 55 миллисекунд).
- Стоимость - произвольное значение, обычно основанное на величине полосы пропускания, денежной стоимости или результате других измерений, которое назначается сетевым администратором.

2.9 Протоколы маршрутизации

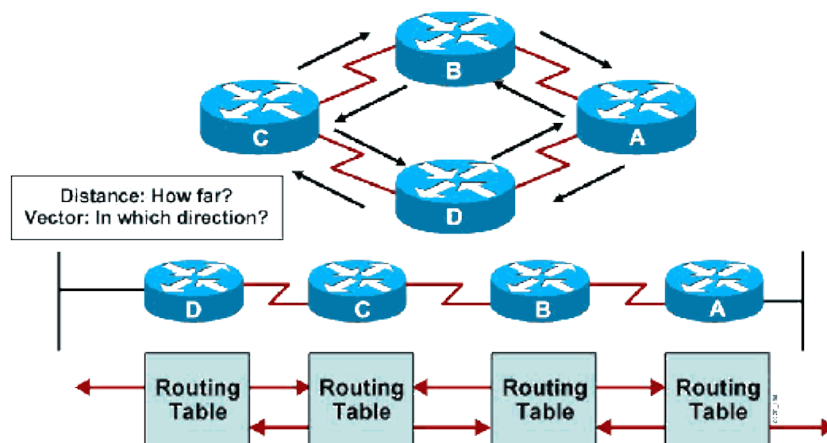
Большинство алгоритмов маршрутизации можно свести к трем основным алгоритмам.

- ✓ Подход на основе маршрутизации по вектору расстояния, в соответствии с которым определяются направление (вектор) и расстояние до каждого канала в сети.
- ✓ Подход на основе оценки состояния канала (также называемый выбором наикратчайшего пути), при котором воссоздается точная топология всей сети (или по крайней мере той части, где размещается маршрутизатор).
- ✓ Гибридный подход, объединяющий аспекты алгоритмов с определением вектора расстояния и оценки состояния канала.

Алгоритм маршрутизации является основой *динамической маршрутизации*. Как только вследствие роста, реконфигурирования или отказа изменяется топология сети, база знаний о сети должна изменяться тоже; это прерывает маршрутизацию.

Необходимо, чтобы знания отражали точное и непротиворечивое представление о новой топологии. В том случае, когда все маршрутизаторы используют непротиворечивое представление топологии сети, имеет место сходимость. Говорят, что сетевой комплекс сошелся, когда все имеющиеся в нем маршрутизаторы работают с одной и той же информацией. Процесс и время, требующиеся для возобновления сходимости маршрутизаторов, меняются в зависимости от протокола маршрутизации. Для сети желательно обладать свойством быстрой сходимости, поскольку это уменьшает время, когда маршрутизаторы используют для принятия решений о выборе маршрута устаревшие знания, и эти решения могут быть неправильными, расточительными по времени или и теми и другими одновременно.

2.10 Алгоритмы маршрутизации по вектору расстояния



4.6.3 Маршрутизации на основе вектора расстояния

Алгоритмы маршрутизации на основе вектора расстояния (также известные под названием алгоритмы Беллмана—Форда (Bellman-Ford algorithms)) предусматривают периодическую передачу копий таблицы маршрутизации от одного маршрутизатора другому. Регулярно посылаемые между маршрутизаторами пакеты актуализации сообщают обо всех изменениях топологии.

Каждый маршрутизатор получает таблицу маршрутизации от своего соседа. Например, на рис. 4.6.3 маршрутизатор В получает информацию от маршрутизатора А. Маршрутизатор В добавляет величину, отражающую вектор расстояния (скажем, количество переходов), которая увеличивает вектор расстояния, и затем передает таблицу маршрутизации своему соседу - маршрутизатору С. Такой же процесс пошагово выполняется между соседними маршрутизаторами во всех направлениях.

Подобным образом алгоритм аккумулирует сетевые расстояния и поэтому способен поддерживать базу данных информации о топологии сети. Однако алгоритмы на основе вектора расстояния не позволяют маршрутизатору знать точную топологию всего сетевого комплекса.

2.11 Алгоритмы маршрутизации с учетом состояния канала связи

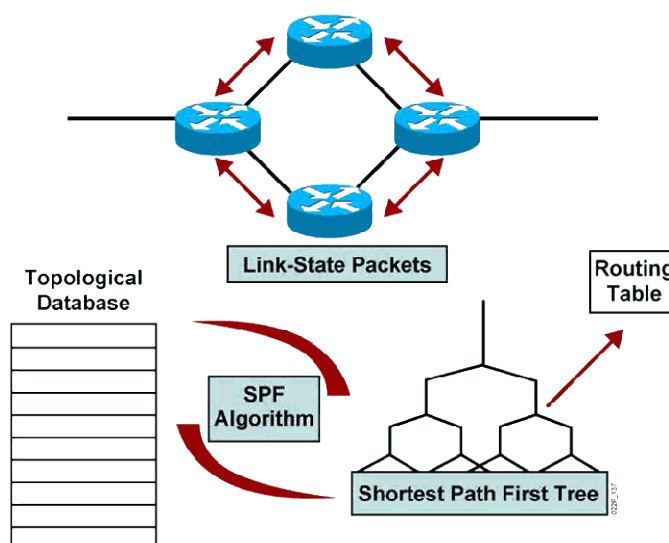


Рис. 4.6.4 Маршрутизация с учетом состояния канала связи

Вторым основным алгоритмом, используемым для маршрутизации, является алгоритм с учетом состояния канала связи. Алгоритмы маршрутизации с учетом состояния канала связи, также известные под названием алгоритмов выбора первого кратчайшего пути (shortest path first (SPF) algorithms), поддерживают сложную базу данных топологической информации. И если алгоритмы с маршрутизацией по вектору расстояния работают с неконкретной информацией о дальних сетях, то алгоритмы маршрутизации с учетом состояния канала собирают полные данные о дальних маршрутизаторах и о том, как они соединены друг с другом.

Для выполнения маршрутизации с учетом состояния канала связи используются сообщения объявлений о состоянии канала (link-state advertisements, LSA), база данных топологии, SPF-алгоритм, результирующее SPS-дерево и таблица маршрутизации, содержащая пути и порты к каждой сети (рис. 4.6.4).

2.12 Сравнение маршрутизации по вектору расстояния с учетом состояния канала связи

Сравнивать маршрутизацию по вектору расстояния и маршрутизацию с учетом состояния канала связи можно в нескольких ключевых областях:

- Процесс маршрутизации по вектору расстояния получает все топологические данные из информации, содержащейся в таблицах маршрутизации соседей. Процесс маршрутизации с учетом состояния канала связи получает широко представление обо всей топологии сетевого комплекса, собирая данные из всех необходимых LSA-пакетов.
- Процесс маршрутизации по вектору расстояния определяет лучший путь с помощью сложения получаемых метрик по мере того, как таблица движется от одного маршрутизатора к другому. При использовании маршрутизации с учетом состояния канала каждый маршрутизатор работает отдельно, вычисляя свой собственный кратчайший путь к пункту назначения.
- В большинстве протоколов маршрутизации по вектору расстояния пакеты актуализации, содержащие сведения об изменениях топологии, являются периодически посылаемыми пакетами актуализации таблиц маршрутизации. Эти таблицы передаются от одного маршрутизатора к другому, что обычно приводит к более медленной сходимости.
- В протоколах маршрутизации с учетом состояния канала связи пакеты актуализации обычно генерируются и рассылаются по факту возникновения изменения топологии. Относительно небольшие LSA-пакеты передаются всем другим маршрутизаторам, что, как правило, приводит к более быстрой сходимости при любом изменении топологии сетевого комплекса.

Маршрутизация по вектору расстояния	Маршрутизация с учетом состояния канала связи
Видит топологию сети глазами соседних маршрутизаторов	Получает общий вид топологии всей сети
Суммирует вектор расстояния от одного маршрутизатора к другому	Вычисляет кратчайший путь до других маршрутизаторов
Частые периодические обновления топологической информации, медленная сходимость	Обновления инициируются фактом изменения топологии; быстрая сходимость
Передаёт копии таблицы маршрутизации только соседним маршрутизаторам	Передаёт пакеты с информацией об актуальном состоянии канала связи всем другим маршрутизаторам

2.13 Алгоритм Беллмана – Форда

Описание алгоритма: Предположим, что вершина 1 является вершиной-входом, и требуется найти длины кратчайших путей от вершины 1 до каждой другой вершины графа. Для этого алгоритма дуговые расстояния могут быть как положительными, так и отрицательными, но не должно быть циклов отрицательной длины. Предположим, что если в графе отсутствует та или иная дуга, то её вес равен бесконечно большому числу. Основная идея алгоритма Беллмана - Форда состоит в том, чтобы сначала найти длины кратчайших путей, при условии, что пути содержат не более двух дуг, не более трех дуг и т.д. Кратчайший путь, при условии, что он содержит не более h дуг, будем называть кратчайшим путем в графе. Согласно идее Беллмана - Форда, Р. Прим предложил алгоритм нахождения длин кратчайших путей, ведущих из произвольной вершины графа во все остальные его вершины (в которые есть пути из вершины-входа), состоящий в присвоении вершинам некоторых потенциалов. Признаком окончания алгоритма является остановка процесса изменения потенциалов.

Алгоритм, предложенный Р. Примом, состоит в выполнении следующих операций:

- 1) Вершине-входу присваиваем потенциал $\varphi_1 = 0$; остальным вершинам – потенциал $\varphi_i = \infty$, $i = 1 \dots n$, где n – число вершин в рассматриваемом графе.
- 2) Для каждой вершины i , смежной с вершиной j из массива вершин L , находим $\varphi_j + l(j, i)$ и проверяем неравенство

$$\varphi_j + l(j, i) < \varphi_i.$$

Если неравенство выполняется, то

$$\varphi_i := \varphi_j + l(j, i).$$

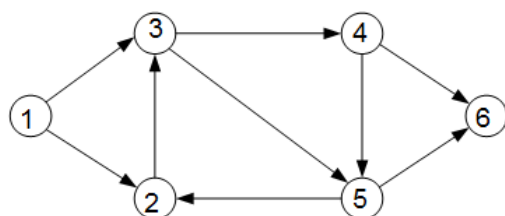
Вершину i заносим в массив L_1 .

- 3) Проверяем массив $L_1 \neq \emptyset$. Если неравенство выполняется $L := L_1$, осуществляем переход к п. 2.
- 4) Конец алгоритма нахождения кратчайших путей.

Пример:

Рассмотрим работу алгоритма на примере.

Дано: Допустим, имеется граф, представленный списком дуг:



(1,2) – 3	(4,6) – 6
(1,3) – 9	(3,5) – 2
(2,3) – 5	(5,2) – -1
(3,4) – 1	(5,6) – 4
(4,5) – 2	(6,0) – 0

Решение:

Результаты работы алгоритма по шагам вычислений сведем в таблицу.

Шаг вычисле- ний	Массив L	Потенциалы вершин						Массив L ₁
		1	2	3	4	5	6	
0	∅	0	∞	∞	∞	∞	∞	1
1	1	0	3	9	∞	∞	∞	2, 3
2	2, 3	0	3	8	10	11	∞	3, 4, 5
3	2, 4, 5	0	3	8	9	10	15	4, 5, 6
4	4, 5, 6	0	3	8	9	10	14	6
5	6	0	3	8	9	10	14	∅

На нулевом шаге производится инициализация согласно п.1 алгоритма. Далее на первом шаге вычислений рассматриваются вершины, смежные с вершиной-входом, потенциалы

рассматриваемых вершин становятся равными длине соединяющей их дуги, если таковое существует, в противном случае – потенциал вершины остается равным ∞ .

На втором шаге рассматриваем вершины, достижимые из вершины-входа при помощи пути, состоящего из двух дуг. Таковыми являются:

- 1) 1 – 2 – 3: потенциал вершины 3 изменяется $\varphi_3 = 8$;
- 2) 1 – 3 – 4: потенциал вершины 4 изменяется $\varphi_4 = 10$;
- 3) 1 – 3 – 5: потенциал вершины 5 изменяется $\varphi_5 = 11$.

Третий шаг:

- 1) 1 – 3 – 4 – 6: потенциал вершины 6 изменяется $\varphi_6 = 16$;
- 2) 1 – 2 – 3 – 5: потенциал вершины 5 изменяется $\varphi_5 = 10$;
- 3) 1 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);
- 4) 1 – 3 – 4 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 12$);
- 5) 1 – 3 – 5 – 6: потенциал вершины 6 изменяется $\varphi_6 = 15$.

Четвертый шаг:

- 1) 1 – 2 – 3 – 5 – 6: потенциал вершины 6 изменяется $\varphi_6 = 14$;
- 2) 1 – 2 – 3 – 4 – 5: потенциал вершины 5 не изменится $\varphi_5 < 11$;
- 3) 1 – 2 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 9$);
- 4) 1 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 15$);
- 5) 1 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 11$).

Пятый шаг:

- 1) 1 – 3 – 5 – 2 – 3 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 17$);
- 2) 1 – 3 – 5 – 2 – 3 – 4: потенциал вершины 4 не изменится ($\varphi_4 < 16$);
- 3) 1 – 3 – 4 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$);
- 4) 1 – 2 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);
- 5) 1 – 2 – 3 – 4 – 5 – 6: потенциал вершины 6 не изменится ($\varphi_6 = 14$);
- 6) 1 – 2 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$).

На пятом шаге вычислений не произошло изменений ни в одном из потенциалов вершин графа, делаем вывод об окончании работы алгоритма.

Восстановим кратчайший путь из вершины 6 в вершину 1. Для этого для каждой вершины j определим такую вершину i , что будет выполняться правило

$$\varphi_i := \varphi_j + l(j, i).$$

Следуя правилу, последовательно находим кратчайший путь:

вершина 6: $i = 5$;

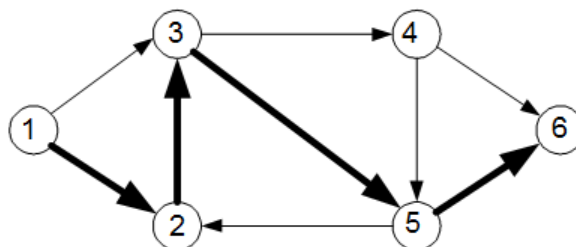
вершина 5: $i = 3$;

вершина 3: $i = 2$;

вершина 2: $i = 1$.

Кратчайшим путем является путь:

1 – 2 – 3 – 5 – 6



3. Задания для выполнения

Дан список дуг с указанием их длин. Необходимо составить по нему рисунок ориентированного графа. Найдите для графа наименьший путь от вершины-входа до вершины с максимальным номером. ($\varphi = 0$)

4. Варианты индивидуальных заданий.

1. $(0;1) - 3, (0;2) - 9, (1;2) - 5,$
 $(2;4) - 1, (1;3) - 8, (2;3) - 2,$
 $(3;5) - 4, (4;5) - 6.$
2. $(0;1) - 4, (0;2) - 5, (1;2) - 8,$
 $(2;4) - 3, (1;3) - 11, (2;3) - 5,$
 $(3;5) - 3, (4;5) - 6.$
3. $(0;1) - 3, (0;2) - 9, (1;2) - 12,$
 $(2;4) - 1, (1;3) - 2, (2;3) - 3,$
 $(3;5) - 10, (4;5) - 5.$
4. $(0;1) - 6, (0;2) - 2, (2;1) - 3,$
 $(2;4) - 6, (1;3) - 1, (2;3) - 5,$
 $(3;5) - 8, (4;5) - 7.$
5. $(0;1) - 6, (0;2) - 5, (1;2) - 1,$
 $(2;4) - 6, (1;3) - 7, (2;3) - 6,$
 $(3;5) - 8, (4;5) - 7.$
6. $(0;1) - 3, (0;2) - 2, (2;1) - 1,$
 $(2;5) - 3, (1;5) - 4, (5;4) - 8,$
 $(5;3) - 5, (3;4) - 3, (4;6) - 2,$
 $(3;6) - 4.$
7. $(0;1) - 10, (0;2) - 5, (2;1) - 4,$
 $(2;5) - 8, (1;5) - 3, (5;4) - 4,$
 $(5;3) - 2, (3;4) - 1, (4;6) - 5,$
 $(3;6) - 7.$
8. $(0;1) - 3, (0;2) - 2, (2;1) - 2,$
 $(2;5) - 12, (1;5) - 8, (5;4) - 2,$
 $(5;3) - 6, (3;4) - 1, (4;6) - 8,$
 $(3;6) - 3.$
9. $(0;1) - 2, (0;2) - 7, (2;1) - 1,$
 $(2;5) - 6, (1;5) - 12, (5;4) - 10,$
 $(5;3) - 5, (3;4) - 4, (4;6) - 2,$
 $(3;6) - 7.$
10. $(0;1) - 4, (0;2) - 2, (2;1) - 1,$
 $(2;5) - 7, (1;5) - 5, (5;4) - 4,$
 $(5;3) - 1, (3;4) - 4, (4;6) - 3,$
 $(3;6) - 7.$
11. $(0;2) - 2, (0;1) - 7, (2;1) - 4,$
 $(2;4) - 9, (1;3) - 3, (3;4) - 1,$
 $(4;6) - 2, (3;5) - 8, (6;5) - 4,$
 $(6;7) - 10, (5;7) - 5.$
12. $(0;2) - 10, (0;1) - 5, (2;1) - 1,$
 $(2;4) - 4, (1;3) - 3, (3;4) - 5,$
 $(4;6) - 3, (3;5) - 10, (6;5) - 10,$
 $(6;7) - 5, (5;7) - 1.$
13. $(0;2) - 4, (0;1) - 6, (2;1) - 4,$
 $(2;4) - 6, (1;3) - 3, (3;4) - 2,$
 $(4;6) - 4, (3;5) - 7, (6;5) - 3,$
 $(6;7) - 8, (5;7) - 5.$
14. $(0;2) - 3, (0;1) - 1, (2;1) - 4,$
 $(2;4) - 2, (1;3) - 6, (3;4) - 4,$
 $(4;6) - 3, (3;5) - 6, (6;5) - 4,$
 $(6;7) - 12, (5;7) - 7.$
15. $(0;2) - 8, (0;1) - 12, (2;1) - 3,$
 $(2;4) - 6, (1;3) - 5, (3;4) - 4,$
 $(4;6) - 10, (3;5) - 4, (6;5) - 6,$
 $(6;7) - 10, (5;7) - 6.$

16. (0;1) – 3, (0;2) – 3, (1;4) – 3,
(2;5) – 3, (1;3) – 2, (0;3) – 4,
(2;3) – 2, (3;4) – 2, (3;6) – 4,
(3;5) – 2, (4;6) – 3, (5;6) – 3.
17. (0;1) – 3, (0;2) – 2, (1;4) – 13,
(2;5) – 13, (1;3) – 7, (0;3) – 11,
(2;3) – 9, (3;4) – 5, (3;6) – 10,
(3;5) – 3, (4;6) – 4, (5;6) – 7.
18. (0;1) – 4, (0;2) – 5, (1;4) – 7,
(2;5) – 7, (1;3) – 5, (0;3) – 8,
(2;3) – 4, (3;4) – 2, (3;6) – 7,
(3;5) – 1, (4;6) – 4, (5;6) – 6.
19. (0;1) – 5, (0;2) – 5, (1;4) – 4,
(2;5) – 5, (1;3) – 3, (0;3) – 10,
(2;3) – 3, (3;4) – 3, (3;6) – 10,
(3;5) – 3, (4;6) – 7, (5;6) – 5.
20. (0;1) – 6, (0;2) – 7, (1;4) – 18,
(2;5) – 19, (1;3) – 8, (0;3) – 14,
(2;3) – 6, (3;4) – 8, (3;6) – 14,
(3;5) – 5, (4;6) – 5, (5;6) – 9.
21. (0;1) – 2, (1;2) – 3, (0;2) – 6,
(2;3) – 4, (3;4) – 2, (2;4) – 7,
(4;5) – 5, (5;6) – 3, (4;6) – 9.
22. (0;1) – 3, (1;2) – 5, (0;2) – 7,
(2;3) – 6, (3;4) – 5, (2;4) – 12,
(4;5) – 3, (5;6) – 2, (4;6) – 4.
23. (0;1) – 5, (1;2) – 4, (0;2) – 10,
(2;3) – 4, (3;4) – 5, (2;4) – 8,
(4;5) – 7, (5;6) – 6, (4;6) – 14.
24. (0;1) – 2, (1;2) – 4, (0;2) – 5,
(2;3) – 4, (3;4) – 5, (2;4) – 8,
(4;5) – 3, (5;6) – 2, (4;6) – 4.
25. (0;1) – 3, (1;2) – 2, (0;2) – 5,
(2;3) – 1, (3;4) – 1, (2;4) – 3,
(4;5) – 2, (5;6) – 2, (4;6) – 3.
26. (0;1) – 10, (0;2) – 4, (1;4) – 8,
(2;5) – 20, (3;1) – 5, (4;3) – 3,
(2;3) – 4, (3;5) – 17, (4;6) – 7,
(5;6) – 5.
27. (0;1) – 6, (0;2) – 2, (1;4) – 4,
(2;5) – 15, (3;1) – 2, (4;3) – 3,
(2;3) – 13, (3;5) – 2, (4;6) – 7,
(5;6) – 1.
28. (0;1) – 2, (0;2) – 3, (1;4) – 1,
(2;5) – 10, (3;1) – 1, (4;3) – 6,
(2;3) – 4, (3;5) – 5, (4;6) – 20,
(5;6) – 6.
29. (0;1) – 2, (0;2) – 3, (1;4) – 7,
(2;5) – 6, (3;1) – 1, (4;3) – 2,
(2;3) – 2, (3;5) – 3, (4;6) – 8,
(5;6) – 10.
30. (0;1) – 3, (0;2) – 7, (1;4) – 8,
(2;5) – 3, (3;1) – 1, (4;3) – 4,
(2;3) – 2, (3;5) – 2, (4;6) – 7,
(5;6) – 6.

5. Контрольные вопросы.

1. Что такое маршрутизатор? Задачи маршрутизации.
2. Статическая и динамическая маршрутизации. Отличия.
3. Алгоритмы маршрутизации. Сравнение.
4. Алгоритм Беллмана-Форда.

6. Список рекомендуемой литературы.

1. В.Г. Олифер, Н.А. Олифер, “Компьютерные сети”.
(ftp://iipo.tu-bryansk.ru/pub/Drozdov/Net/Books/book_comp_net_olifer.chm)
2. Компьютерные сети. 4-е изд. / Э. Таненбаум. – СПб.: Питер, 2006. – 992 с.: ил. – (Серия “Классика computer science”).
(<ftp://iipo.tu-bryansk.ru/pub/Drozdov/Net/Books/TanenbaumKompjuternyeSeti.djvu>)

ЛАБОРАТОРНАЯ РАБОТА №5 ИССЛЕДОВАНИЕ ПРОТОКОЛА HTTP

1. Цель работы

Изучить основы работы с протоколом HTTP, telnet.

2. Общие сведения

2.1 Telnet

Telnet - сетевой протокол для реализации текстового интерфейса по сети. Название «**telnet**» имеет также утилита, реализующая клиентскую часть протокола. Исторически **telnet** служил для удалённого доступа к интерфейсу командной строки операционных систем. Протокол **telnet** может использоваться для выполнения отладки других протоколов на основе транспорта TCP.

Утилита **telnet** поддерживает следующие команды:

- **Close** – закрытие текущего подключения.
- **Display** – отображение параметров операции.
- **Open** – подключение к сайту.
- **Quit** – выход из telnet.
- **Set** – установление параметров.
- **Send** – отправление строки на сервер.
- **Status** – вывод сведений о текущем состоянии.
- **Unset** – сброс параметров.

Используя утилиту **telnet** можно, например, вручную отправить запрос клиента и получить ответ сервера по протоколу HTTP.

Для этого выполним следующую последовательность действий:

1. Запуск утилиты telnet
2. Установление соединения с веб-сервером с помощью команды:
open имя_хоста 80
3. Формирование запроса клиента
4. Получение ответа сервера

Включение telnet

1. Нажмите Start, затем выберите Control Panel.
 2. Нажмите Programs and Features.
 3. Нажмите Turn Windows features on or off.
 4. Отметьте чекбокс напротив Telnet Client.
 5. Нажмите ОК.
 6. Начнется процесс установки.
- Когда он закончится, клиент установлен и готов к работе.

```
C:\>telnet
Welcome to Microsoft Telnet Client

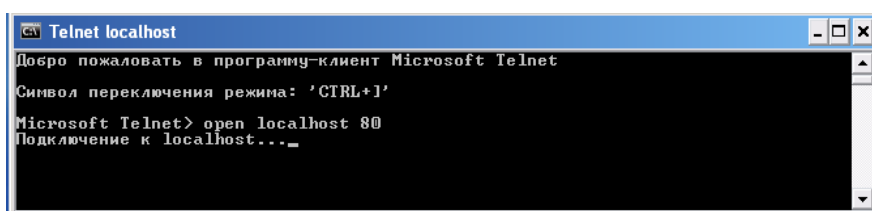
Escape Character is 'CTRL++'
```

```
Microsoft Telnet> set localecho
Local echo on
Microsoft Telnet> open www.example.com 80
Connecting To www.example.com...
GET / HTTP/1.0
```

Пример

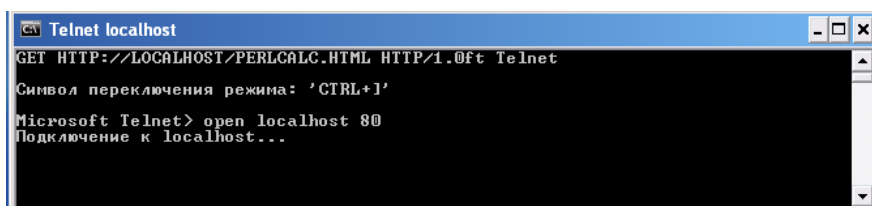
1. Устанавливаем соединение:

open localhost 80

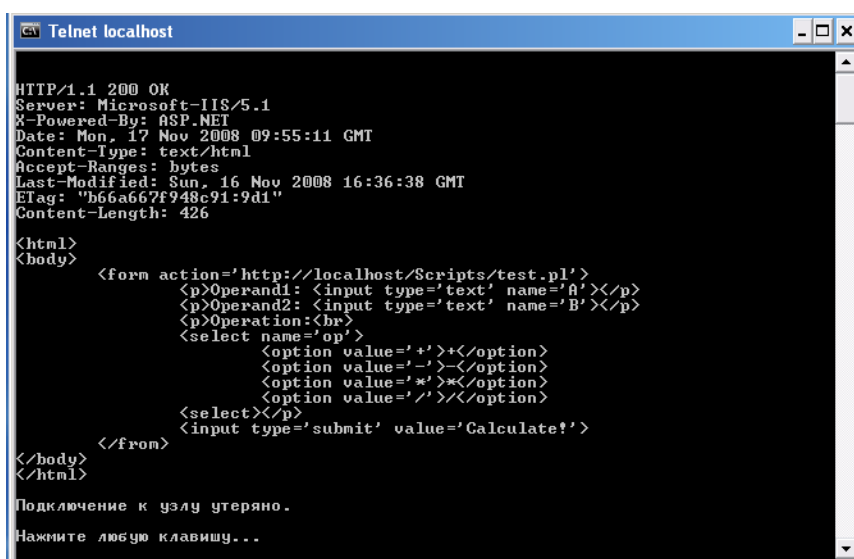


2. Формируем строку состояния запроса клиента:

GET HTTP://LOCALHOST/PERLCALC.HTML HTTP/1.0 <ENTER><ENTER>



3. Получаем ответ сервера:



Видно, что ответ веб-сервера *localhost* содержит строку состояния (с кодом успешного завершения 200), поля заголовка (*Server*, *Date*, *Content-type* и др.) и тело, содержащее HTML код запрошенного клиентом документа *http://localhost/perlcalc.html*.

Задание. Ознакомление с протоколом HTTP с помощью утилиты telnet.

- 1) Запустите сеанс *telnet* (запускается в командной строке командой *telnet*). При этом появится подсказка *MicrosoftTelnet>*. С полным списком команд можно ознакомиться с помощью команды *help*.
- 2) Разрешите режим отображения вводимых с клавиатуры символов с помощью команды *set local echo*.
- 3) В соответствии с протоколом HTTP необходимо установить соединение с веб-сервером. Для этого с помощью команды *open* устанавливается соединение, например: *openwww.yandex.ru 80*.
- 4) Сформируйте клиентский запрос. Как минимум он должен содержать строку состояния, например: *GET HTTP://WWW.YANDEX.RU/INDEX.HTML HTTP/1.0*

Если поля запроса отсутствуют, то ввод заканчивается двумя нажатиями клавиши <ENTER> для вставки пустой строки после заголовка.

2.2 HTTP (HyperText Transfer Protocol)

HTTP (HyperText Transfer Protocol - протокол передачи гипертекста) был разработан как основа World Wide Web.

Работа по протоколу HTTP происходит следующим образом: программа - клиент устанавливает TCP-соединение с сервером (стандартный номер порта- 80) и выдает ему HTTP-запрос. Сервер обрабатывает этот запрос и выдает HTTP-ответ клиенту.

Структура HTTP-запроса

HTTP-запрос состоит из заголовка запроса и тела запроса, разделенных пустой строкой. Тело запроса может отсутствовать.

Заголовок запроса состоит из главной (первой) строки запроса и последующих строк, уточняющих запрос в главной строке. Последующие строки также могут отсутствовать.

Запрос в главной строке состоит из трех частей, разделенных пробелами: Метод (иначе говоря, команда HTTP):

- **GET** - запрос документа. Наиболее часто употребляемый метод
- **HEAD** - запрос заголовка документа. Отличается от GET тем, что выдается только заголовки запроса с информацией о документе. Сам документ не выдается.
- **POST** - этот метод применяется для передачи данных CGI-скриптам. Сами данные следуют в последующих строках запроса в виде параметров.
- **PUT** - разместить документ на сервере.

Ресурс - это путь к определенному файлу на сервере, который клиент хочет получить (или разместить - для метода PUT). Если ресурс - просто какой-либо файл для считывания, сервер должен по этому запросу выдать его в теле ответа. Если же это путь к какому-либо CGI-скрипту, то сервер запускает скрипт и возвращает результат его выполнения.

Версия протокола - версия протокола HTTP, с которой работает клиентская программа.

Таким образом, простейший HTTP-запрос может выглядеть следующим образом:

GET / HTTP/1.0

Здесь запрашивается корневой файл из корневой директории web-сервера.

Строки после главной строки запроса имеют следующий формат:

Параметр: значение

Таким образом задаются параметры запроса. Это является необязательным, все строки после главной строки запроса могут отсутствовать; в этом случае сервер принимает их значение по умолчанию или по результатам предыдущего запроса (при работе в режиме Keep-Alive).

Наиболее употребительные параметры HTTP-запроса:

- ✓ **Connection** (соединение) - может принимать значения Keep-Alive и close. Keep-Alive ("оставить в живых") означает, что после выдачи данного документа соединение с сервером не разрывается, и можно выдавать еще запросы. Большинство браузеров работают именно в режиме Keep-Alive, так как он позволяет за одно соединение с сервером "скачать" html-страницу и рисунки к ней. Будучи однажды установленным, режим Keep-Alive сохраняется до первой ошибки или до явного указания в очередном запросе Connection: close ("закрыть") - соединение закрывается после ответа на данный запрос.
- ✓ **User-Agent** - значением является "кодовое обозначение" браузера, например: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)
- ✓ **Accept** - список поддерживаемых браузером типов содержимого в порядке их предпочтения данным браузером, например:
*Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */**
- ✓ **Referer** - URL, с которого перешли на этот ресурс.
- ✓ **Host** - имя хоста, с которого запрашивается ресурс. Необходимо, если на сервере имеется несколько виртуальных серверов под одним IP-адресом. В этом случае имя виртуального сервера определяется по этому полю.
- ✓ **Accept-Language** - поддерживаемый язык. Имеет значение для сервера, который может выдавать один и тот же документ в разных языковых версиях.

Формат HTTP-ответа

Формат ответа очень похож на формат запроса: он также имеет заголовок и тело, разделенное пустой строкой.

Заголовок также состоит из основной строки и строк параметров, но формат основной строки отличается от таковой в заголовке запроса.

Основная строка запроса состоит из 3-х полей, разделенных пробелами:

1. Версия протокола - аналогичен соответствующему параметру запроса.
2. Код ошибки - кодовое обозначение "успешности" выполнения запроса. Код 200 означает "все нормально" (OK).
3. Словесное описание ошибки - "расшифровка" предыдущего кода. Например для 200 это OK, для 500 - Internal Server Error.

Наиболее употребительные параметры http-ответа:

- ✓ **Connection** - аналогичен соответствующему параметру запроса. Если сервер не поддерживает Keep-Alive (есть и такие), то значение Connection в ответе всегда close.
- ✓ **Content-Type** ("тип содержимого") - содержит обозначение типа содержимого ответа. В зависимости от значения Content-Type браузер воспринимает ответ как HTML-страницу, картинку gif или jpeg, как файл, который надо сохранить на диске, или как что-либо еще и предпринимает соответствующие действия. Значение Content-Type для браузера аналогично значению расширения файла для такой системы как Windows.

Некоторые типы содержимого:

- **text/html** - текст в формате HTML (веб-страница);

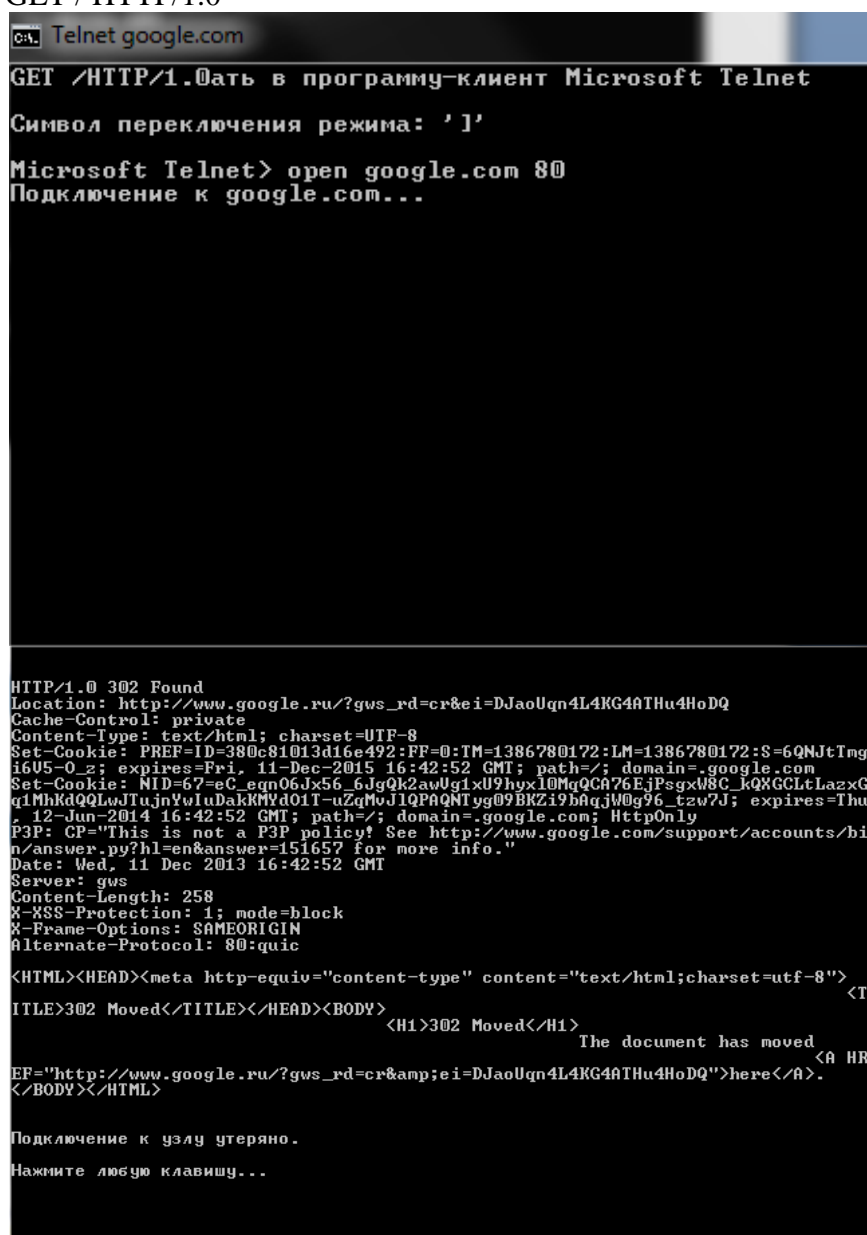
- **text/plain** - простой текст (аналогичен "блокнотовскому");
 - **image/jpeg** - картинка в формате JPEG;
 - **image/gif** - картинка в формате GIF;
 - **application/octet-stream** - поток "октетов" (т.е. просто байт) для записи на диск.
- ✓ **Content-Length** ("длина содержимого") - длина содержимого ответа в байтах.
- ✓ **Last-Modified** ("Модифицирован в последний раз") - дата последнего изменения документа.

Примеры использования утилиты telnet

Соединение с веб-узлом, метод GET HTTP-запроса:

Open google.com 80

GET / HTTP/1.0



```
Telnet google.com
GET /HTTP/1.0
Символ переключения режима: ']'
Microsoft Telnet> open google.com 80
Подключение к google.com...

HTTP/1.0 302 Found
Location: http://www.google.ru/?gws_rd=cr&ei=DJaoUqn4L4KG4ATHu4HoDQ
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie: PREF=ID=380c81013d16e492:FF=0:TM=1386780172:LM=1386780172:S=6QNJtImg16U5-0_z; expires=Fri, 11-Dec-2015 16:42:52 GMT; path=/; domain=.google.com
Set-Cookie: NID=67=eG_eqn06Jx56_6JgQk2auUg1xU9hyx10MqQCA76EjPsgxW8C_kQXGCLtLazxGq1MhKdQQLwJtuJnYwluDakMYd01I-uZqHvJlQPAQNIyg07BKZi7baqjW0g96_tzw7J; expires=Thu, 12-Jun-2014 16:42:52 GMT; path=/; domain=.google.com; HttpOnly
P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Date: Wed, 11 Dec 2013 16:42:52 GMT
Server: gws
Content-Length: 258
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic

<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8"><TITLE>302 Moved</TITLE></HEAD><BODY>
  <H1>302 Moved</H1>
  The document has moved
  EF="http://www.google.ru/?gws_rd=cr&ei=DJaoUqn4L4KG4ATHu4HoDQ">here</A>.
</BODY></HTML>

Подключение к узлу утеряно.
Нажмите любую клавишу...
```

Open ya.ru 80

GET / HTTP/1.0

```
Сл. Telnet ya.ru
GET /HTTP/1.0ать в программу-клиент Microsoft Telnet
Символ переключения режима: ']'
Microsoft Telnet> open ya.ru 80
Подключение к ya.ru...

div><div class="b-domik_username"><span class="b-form-input b-form-input theme_
grey b-form-input_size_m b-domik_input i-bem" onclick="return <'b-form-input':{
}>"><span class="b-form-input_hint-wrap"><label class="b-form-input_hint"for="
id001"><span class="b-form-input_box"><input class="b-
form-input_input" name="login" id="id001" value="" tabindex="3" /></span></span>
</div><div class="b-domik_password"><span class="b-form-input b-form-input theme_
grey b-form-input_size_m b-domik_input i-bem" onclick="return <'b-form-input':
{}>"><span class="b-form-input_hint-wrap"><label class="b-form-input_hint" fo
r="id002"><span class="b-form-input_box"><input clas
s="b-form-input_input" name="passwd" value="" id="id002" tabindex="4" type="pas
sword"/></span><div class="b-domik_lock"></div></span></div><div class="b-domik_
haunter"><span class="b-form-checkbox b-form-checkbox theme_grey-1 b-form-chec
kbox_size_1 b-domik_checkbox i-bem" onclick="return <'b-form-checkbox':{}>"><sp
an class="b-form-checkbox_inner"><input name="twoweeks" value="yes" type="hidde
n"/><input class="b-form-checkbox_checkbox" id="b-domik-haunter" tabindex="5"
name="twoweeks" value="no" autocomplete="off" type="checkbox" /><i class="b-form-
checkbox_bg"><i class="b-form-checkbox_tick"></i></i></span><label class="b-f
orm-checkbox_label" for="b-domik-haunter"></label>
</span></div><div class="b-domik_button"><span class="b-form-button b-form-butto
n theme_grey-no-transparent-m b-form-button_size_m i-bem" onclick="return <'b-
form-button':{name:'b-form-button'}>"><i class="b-form-button_left"></i><span c
lass="b-form-button_content"><span class="b-form-button_text"></span>
</span><input type="submit" hidefocus="true" tabindex="6" class="b-form-button_
input" value="" /></span></div><div class="b-domik_social"><div class="b-domik_
i-bem" onclick="return <'b-domik_social':{size:16},'b-fade':{target:'b-domi
k'}>"></div><div class="b-domik_remember"><a class="b-link b-domik_remind" h
ref="https://passport.yandex.ru/passport?mode=restore" tabindex="7">
</a><div class="b-topbar-toggler"><span class="b-link i-bem
b-link_pseudo_yes b-link_inner_yes"onclick="return<'b-link':{}>"><s
pan class="b-link_inner"></span></div></div></div><div class="b-popupa_
shadow"><div></div></div></div><div class="b-wdgt-control"><div class="b-
widget_control"><a id="b-wdgt-setup_button" class="b-widget_control_setup" h
ref="?edit=1" title=""></div><div class="b-wdgt-send_button"
class="b-widget_control_send" href="?edit=1" title=""></div><div class="b-wdgt-del_button" class="b-widget_control_del" href="?edi
t=1" title=""></div></div></div><div><script src="//mc.yandex.ru/
metrika/watch.js" type="text/javascript"></script><script type="text/javascript"
>try { var yaCounter722545 = new Ya.Metrika({id:722545, accurateTrackBounce: 100
00}); catch(e) {}</script><div></div></div><div><div class="b-
http://yabs.yandex.ru/count/B90toG4YGF40W0g0A0ZhfPvLe5KP6yq4ba1fE03AxmC25gaUr4x
GHZ0m00=d0-wqwcYw00NbGAR1cbQYgzU5RcbhXxAYBe9Ph03fZiGe1y2th41lJe9Ph0J061x1G00?wmo
de=0" style="display:none; position:absolute;" /></div></div><div><div class="i-flashcookie
i-flashcookie_type_inline g-js" onclick="return {name:'i-flashcookie'}"><span st
yle="background:url(http://kiks.yandex.ru/fu/) no-repeat;"></span></div><div><div class="b-counter" style="background: url(http://www.tns-counter.ru/U13a**8c82e34
8ed889a1c3e540e846379127b**yandex_ru/ru/CP1251/tmsec=yandex_main/0);"></div></div>
5.wfront.yandex.net--<script>staticVersion='1.746';$<function(){$<var clock = new
MordaClock();clock.locale = {}></div></div><div><div class="b-datet
ime_date"><span class="b-datetime_day"></span><span class="b-datetime_
_month"></span></div></div>
Подключение к узлу утеряно.
Нажмите любую клавишу...
```

Соединение с веб-узлом, метод HEAD HTTP-запроса:

Open ya.ru 80
HEAD / HTTP/1.0

```
ca. Telnet ya.ru
HEAD /HTTP/1.0ть в программу-клиент Microsoft Telnet
Символ переключения режима: ']'
Microsoft Telnet> open ya.ru 80
Подключение к ya.ru...
```

```
HTTP/1.1 200 Ok
Server: nginx
Date: Wed, 11 Dec 2013 16:52:04 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 170818
Connection: close
Cache-Control: no-cache,no-store,max-age=0,must-revalidate
Expires: Wed, 11 Dec 2013 16:52:05 GMT
Last-Modified: Wed, 11 Dec 2013 16:52:05 GMT
P3P: policyref="/w3c/p3p.xml", CP="NON DSP ADM DEU PSD IUDo OUR IND STP PHY PRE
NAU UNI"
Set-Cookie: yandexuid=3234894221386780725; Expires=Sat, 09-Dec-2023 16:52:04 GMT
; Domain=.yandex.ru; Path=/
X-Frame-Options: DENY
X-XRDS-Location: http://openid.yandex.ru/server_xrds/

Подключение к узлу утеряно.
Нажмите любую клавишу...
```

Соединение с веб-узлом, метод GET HTTP-запроса, запрос картинки с сервера:

Open www.makak.ru 80
GET /wp-content/uploads/2010/09/telnet-dsl-dlink-6.jpg

```
Сл. Telnet www.makak.ru

GET /wp-content/uploads/2010/09/telnet-dsl-dlink-6.jpg /HTTP/1.0

Символ переключения режима: ']'

Microsoft Telnet> open www.makak.ru 80
Подключение к www.makak.ru...

н FbWC^~eYVH3UMФaCyjn\|^д^щ$|N<Г-Ч>+т Ce Ea Ece-EJm''EИь O||X /|;aип || +r!!пт5
OщЧббшйп||мВкAmX6CШLCPкт*а?|риф*|мzпMи|| YE |лuc*Δ+|E|NH+цUX |Y||Им<т1Жс+| dyГ+б|| E
|;вш|||Θгш>PЦд;yo*пфру-`|μb+0-Г|E|х\х|√3√1a Eп9э<Я| -3-/+γ<o=H|| гЩ\▼H&μ/|▼с+оФб|| Eтb
B^д
q>_||оKΘSNтpυBGqtQEηQEηQEjxcZЧ|Г'Y0nmms=л±O;д/Δь> ΓPNA±◆
||тн
iW>т||пГ1ГгЫГ`|уу$υbE;||х|M-sz*жн*и±Яb^д|| E|нSЯ|3<\M0г9||zYIMz=|MëГ||±O
YUEIc1BCγYC>π || 2A^zuwyM°Яe°T
: |нIGZ
> |нB>OЩ|фрмπ>±·Tnm E7ëB^±=gИпно<Sc? xX||;M&±кZ|| пC-μ/Ш9Kбф|| BAμJ90b-|| ca^Xμi|E|Yг||
|BÛ
;лИ~^кЖВТ<ДЖΘIP±9|E||гy+·. AÛH|e-η
bMfo!BOμ+*Θ,пQ_ЦьMIπ^дeμГю^д%L||д&R%e$||A°oy*БЮкHμ||±ëцПs*й±Y||R>|<>liX#|||▼+gi9
1μжIS EбKтшBк±Eu*оХтн'Q E тп|б;||F+
с±
Θe-P±s|s||0>±±-т/±CдXμ>▼·ьдG±д||0,кэ,|♥|||P±EΔHРг||Y 5-Z
[π°4μUμπ||L XтI|>,±
<|xÿ|Г=Жлап||нq&Чс>7QKс±цдMЦУБ<Й Pн!U<тQД_p
0±γЦх0;γLπPToQBr||тDЮ/ë|||с>^шÛ0iπ±7G-6π,аЮы±BЫxΔ1Ш±я\>NдсN0x6e
K|të<o&Й&πс>f|хЕγ||b^S7!гч+Я?ц√^yυlEicя°X~
μγo fff,;b||=зYÛγ&b+γCπям·||π;E<♥πi±πMлш
`Xн||иY^H S±YБ*W^тш||жЯB, E·k
U||E|||юK-9П*ji± ±*ÛЯJY
x5|ИГн-Q|хP|17Ф||π||π||сγ|ш7|4μ4лγ±|||Ûs|-μ±±3·т-<T8gf||бHКγ/||Г-||ш||ÛБ
юY
▼^пх|Pπг*Б|E'мHсÛPp|'e-ewZ||бiγ:f3эA1мкЦq0.Y||Y||J±±Q71ч>гд^щ$|N<Г-Ч>+т Ce Ea Ece-EJm
кπз^шЦ.эX||л|X>УФПBrë±с||п|=BхH: гM||озγLмр!e*Ц||\·e^б||тY$^kU:еБ|||Y fCмннГэб|hν,
I8ь±бbÛ±
b||;EB''h±±тt0ь
<х$*п>·.o+6C>дEtUбYTEн||=N F|ы+γH;°хнЙдEH||УπXpTsÛKq||б√||^||=E
=G my ;=14=ΘE||CO||±бв гP
±2H$Ж|х8♥ μ||ëйнкΔμ||/GCTÛt||μePμπ1g
grX||γ||γнπ◆F!↓-K=#Q±4x4γ
тЛμ6UP=μ±>яШB8м=7P|юO||>^х;Q||1/!M&Й#°KцhΔ!||бтγт*шП>μ=Ûπq^дт\||т||hγd||πноDR*гь7p|||
π||
Û||,ТH±^Ц±?H!Q7лuf Eя1,мнд|||Э%пШ<μ*кФ;|| μSb||?EηΘ;μЛS-RI
NM^b>
p||;H8=8-|Л<W'Ймь#||π|ы||
B
;X||επ, <±||-eg8||^|||<♥с>т> ||EπИм5Э>8$|||μgRPμFC=ÛнM/|||±ЦЮ4[5±CсÛЭнш|IPL^CdΓ^±dqBΘь
ьФша||
|PYJΓ&||T3/б±*МPтPЕГ5=±e+фПтM3|ГЭJлс=;QпF-ЫтOω-ë`E^μ BAРπ|ΘЗРхк·ПЙ||УМ''πQμπ±LXYUИГ
WNπг||±=|X4±hш||д|±f f||ÛЮкЧвA7<CХuγoэ±цWЛЦ||^МО>H фTЧ>3||б|
Подключение к узлу утеряно.
```


3. Задания для выполнения

С помощью программы TELNET осуществить взаимодействие по протоколу HTTP с несколькими web-ресурсами, находящимися на разном "расстоянии" от нас: в локальной сети ЗНУ, в украинском сегменте Интернет, в "мировом" интернете. В форме запроса клиента применить следующие опции:

5. запрос обычного html документа;
6. запрос изображения с сервера;
7. запрос pdf документа;
8. запрос с передачей параметров по методу GET (например запрос на поисковый сервер)
 - а. запрос с передачей параметров по методу POST (например запрос на авторизацию)

Для связи с сервером по протоколу http использовать программу TELNET.

По окончании работы необходимо подготовить отчет. В отчете привести тексты запросов клиента и ответов сервера.

4. Варианты индивидуальных заданий.

Смотрите п. 3.

5. Контрольные вопросы.

1. Что такое Telnet?
2. Основные команды telnet.
3. Формат http-заголовка.
4. Методы HTTP-запроса.
5. Параметры HTTP-запроса.
6. Формат HTTP-ответа.
7. Основные параметры HTTP-ответа.
8. Типы содержимого HTTP-ответа.

ЛАБОРАТОРНАЯ РАБОТА №6 УСТАНОВКА И АДМИНИСТРИРОВАНИЕ WEB-СЕРВЕРА

1. Цель работы

Изучить основы web-сервера, а также установку и настройку.

2. Общие сведения

2.1 Основные понятия

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-поток или другими данными. Веб-серверы - основа Всемирной паутины.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и компьютер, на котором это программное обеспечение работает.

Клиенты получают доступ к веб-серверу по URL адресу нужной им веб-страницы или другого ресурса.

Дополнительные функции

Дополнительными функциями многих веб-серверов являются:

- ✓ ведение журнала сервера про обращения пользователей к ресурсам;
- ✓ аутентификация пользователей;
- ✓ поддержка динамически генерируемых страниц;
- ✓ поддержка HTTPS для защищенных соединений с клиентами.

Программное обеспечение

На сегодня наиболее распространёнными веб-серверами являются:

- Apache - веб-сервер с открытым исходным кодом, наиболее часто используется в Unix-подобных ОС;
- IIS от компании Microsoft, распространяется с ОС семейства Windows NT Клиенты.

В качестве клиентов для обращения к веб-серверам могут использоваться совершенно различные устройства:

- веб-браузер - самый распространенный способ;
- специальное программное обеспечение может самостоятельно обращаться к веб-серверам для получения обновлений или другой информации;
- мобильный телефон может получить доступ к ресурсам веб-сервера при помощи протокола WAP;
- другие интеллектуальные устройства или бытовая техника.

Apache HTTP-сервер

Apache HTTP-сервер — это веб-сервер, с открытым исходным кодом. С апреля 1996 это самый популярный HTTP-сервер в Интернете. По данным компании Netcraft (<http://www.netcraft.com/Survey/>) общее число web-узлов, работающих под его управлением, к концу 1998 г. достигало 2 млн (55 % общего числа узлов), в мае 1999 года - 57 % веб-серверов, в августе 2004 - 67 %, и это число постоянно растет. Для сравнения: на долю серверов Microsoft приходится 25 %, Netscape – 7 %.

Будучи бесплатной и открытой программой, предназначенной для бесплатных же Unix-систем (FreeBSD, Linux и др.), **Apache** по функциональным возможностям и надежности не уступает коммерческим серверам, а широкие возможности конфигурирования позволяют настроить его для работы практически с любой конкретной системой. Существуют локализации сервера для различных языков, в том числе и для русского. Также основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д. Поддерживает IPv6.

Недостатком наиболее часто называется отсутствие удобного стандартного интерфейса для администратора.

2.2 Администрирование и установка WEB-СЕРВЕРА

Список директив (настроек), используемых для конфигурации web-сервера

Каждая строка конфигурационного файла, кроме строк, начинающихся с #, описывает какую-то директиву конфигурации и ее значение. Строки, начинающиеся с # – это комментарии.

Описание директив, необходимых для выполнения лабораторной работы, приведено в табл 1.

Таблица 1

Описание основных директив конфигурирования web-сервера

Наименование директивы	Описание
ServerType standalone	Показывает тип запуска программы <i>httpd</i> – она может вызываться через <i>inetd</i> (<i>inetd</i>) или как отдельный демон (<i>standalone</i>). По умолчанию вызывается как отдельный демон
Port 80	Порт, по которому http-сервер будет получать запросы. По умолчанию 80
HostnameLookups off	Определяет, будут ли записываться в лог-файлы имена (<i>on</i>) или только <i>ip</i> -адреса (<i>off</i>) хостов, просматривающих сайт. По умолчанию <i>off</i>
User nobody Group nobody	Определяет то, под каким пользователем будет запускаться <i>httpd</i> . Для доступа к 80 порту <i>httpd</i> должен запускаться под <i>root</i> 'ом. Если в директивах User и Group указаны другие пользователи, то <i>httpd</i> все свои действия выполняет с правами этих пользователей, а не <i>root</i> 'а
ServerAdmin root@localhost	Здесь указывается <i>e-mail</i> администратора, отвечающего за работу сервера
ServerRoot /etc/httpd	Каталог, в котором находятся все конфигурационные файлы, лог-файлы и модули. Все пути, которые используются далее, указаны относительно этого каталога
ErrorLog logs/error_log	Местонахождение файла логов об ошибках сервера

LogLevel warn	Уровень сообщений в файлах логов. Возможные значения: <i>debug, info, notice, warn, error, crit, alert, emerg</i>
LoadModule имя_модуля modules/имя_файла_модуля.so	Команда используется для загрузки различных модулей сервера Apache
ClearModuleList	Очистка списка модулей
AddModule имя_файла_модуля.c	Добавления модулей в список модулей
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined LogFormat "%h %l %u %t \"%r\" %>s %b" common LogFormat "%{Referer}i -> %U" referer LogFormat "%{User-agent}i" agent LogFormat "формат" имя	Описание различных форматов записи логов. Имя формата используется в директиве <i>CustomLog</i>
CustomLog logs/access_log common	Задаёт формат файла логов запросов. С помощью <i>CustomLog</i> можно задать файл логов любого формата и имени
PidFile /var/run/httpd.pid	Имя файла, в котором хранится <i>PID</i> запущенного <i>httpd</i> . Используется для останова сервера и перезапуска
UseCanonicalName on	Разрешает использование коротких имен в <i>URL</i> 'ах. Удобно при использовании в Intranet-серверах. Если включено (<i>on</i>), преобразует, например строку <i>http://www/stat</i> в строку <i>http://www.твой-домен.ru/stat</i> , если отключено (<i>off</i>), преобразования не происходит
Timeout 300	Время в секундах, по истечении которого при отсутствии запросов сервер прерывает связь с клиентом
KeepAlive On	Разрешать (<i>on</i>) или нет(<i>off</i>) множественные запросы через одно TCP соединение
MaxKeepAliveRequests 100	Максимальное количество запросов через одно TCP соединение
KeepAliveTimeout 15	Время ожидания следующего запроса в секундах
MinSpareServers 8 MaxSpareServers 20	Минимальное и максимальное значение загруженных процессов <i>httpd</i> . Если набрать в консоли ps ax grep httpd , то можно посмотреть все процессы <i>httpd</i> , загруженные на машине. Если используется Apache на локальной машине, то необходимо уменьшить значение <i>MinSpareServers</i>
StartServers 10	Количество изначально запускаемых процессов
MaxClients 150	Ограничивает количество одновременно подключенных клиентов к серверу

MaxRequestsPerChild 100	Максимальное количество запросов, которые обрабатывает процесс до завершения работы
DocumentRoot /home/httpd/html	Путь к файлам сайта. Здесь хранятся те файлы, которые пользователь получает при наборе URL'a <i>http://мой.сайт.ru</i> . После установки там находится документация по Apache . Эти файлы следует заменить на файлы своего сайта
UserDir public_html	Имя каталога в домашней директории пользователя, где должны находиться файлы его веб-страницы. Обращение к ним происходит по URL'у <i>http://мой.сайт.ru/~имя_пользователя</i> (обрати внимание на ~ ("тильда"))
DirectoryIndex index.html index.shtml index.cgi	Файлы индекса каталога. Это те файлы, которые выводятся при задании в браузере URL'a, являющемся ссылкой на каталог. Например, <i>http://localhost/manual</i> – выведет файл <i>index.html</i> , находящийся в каталоге <i>/home/httpd/html/manual</i> . Если файл индекса отсутствует в каталоге, выводится все содержимое каталога. Можно задать <i>DirectoryIndex default.htm</i>
FancyIndexing on	При включении этой опции список файлов каталога будет выводиться в виде таблицы, в которой над каждой колонкой есть ссылка, при нажатии на которую список сортируется по этой колонке (например, по времени модификации или размеру)
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip AddIconByType (TXT,/icons/text.gif) text/* ... AddIcon /icons/binary.gif .bin .exe ... DefaultIcon /icons/unknown.gif	Эти директивы задают имя файла иконки в зависимости от типа, расширения и имени файла, которая будет выводиться перед именем файла в списке содержимого каталога
ReadmeName README HeaderName HEADER	<i>ReadmeName</i> – имя README файла, который сервер ищет по умолчанию. <i>HeaderName</i> – имя файла, который включается в листинг каталога в качестве заголовка.
IndexIgnore .??* *~ *# HEADER* README* RCS	Файлы, не включаемые в список каталога
AccessFileName .htaccess	Имя файла с правилами доступа, содержимое которого проверяется для каждого каталога
TypesConfig /etc/mime.types	Файл с описаниями типов файлов и принадлежащих к ним расширений
DefaultType text/plain	Тип файла по умолчанию для тех файлов, тип которых неизвестен

AddEncoding x-compress Z AddEncoding x-gzip gz	Позволяет некоторым браузерам распаковывать “сжатые” и gzip-овые архивы "на лету", т. е. при копировании файла типа <i>textcounter.tar.gz</i> получается уже распакованный файл <i>textcounter.tar</i>
AddLanguage en .en AddLanguage fr .fr AddLanguage de .de AddLanguage da .da AddLanguage el .el AddLanguage it .it	Позволяет определить язык документа. Если на сайте есть английская и русская версии страницы, то можно создать файлы <i>doc.html.ru</i> и <i>doc.html.en</i> и в зависимости от языка, прописанного в браузере клиента, ему будет показываться одна из страниц при обращении к <i>doc.html</i> . Расширение не обязательно должно совпадать с аббревиатурой языка
LanguagePriority en fr de	Определяет приоритет языка документа
Alias /icons/ /home/httpd/icons/	Разрешает создавать <i>алиасы</i> (альтернативные имена) для каталогов, т.е. если html-страница лежит в <i>/home/httpd/html</i> , то при обращении к <i>http://мой.сайт.ru/icons</i> Apache будет искать каталог <i>/home/httpd/html/icons</i> , а после определения такого <i>алиаса</i> Apache будет искать каталог <i>/home/httpd/icons</i>
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/	То же самое, что и <i>Alias</i> , но для серверных скриптов (CGI)
AddType text/html .shtml AddHandler server-parsed .shtml	Добавляет поддержку SSI для файлов с расширением .shtml
AddHandler imap-file map	Добавляет поддержку <i>imagemap</i> 'ов
BrowserMatch "Mozilla/2" nokeepalive BrowserMatch "MSIE 4\0b2;" nokeepalive downgrade-1.0 force-response-1.0 BrowserMatch "RealPlayer 4\0" force-response-1.0 BrowserMatch "Java/1\0" force-response-1.0 BrowserMatch "JDK/1\0" force-response-1.0	Прописывает различные параметры работы <i>httpd</i> при обращении к нему описанных браузеров
<Directory /> Options None AllowOverride None </Directory>	Описывает параметры доступа к корневой директории. Options контролирует то, какие опции доступа к директории доступны. Как видно из примера – никакие (None). Следовательно, доступа никакого. AllowOverride None – запрещает изменять опции доступа посредством содержимого файла <i>.htaccess</i> каталога
<Directory /home/httpd/html> Options Indexes Includes FollowSymLinks AllowOverride None order allow,deny	Устанавливает параметры доступа к каталогу <i>/home/httpd/html</i> . Опции доступа (Options) следующие:

<i>allow from all</i> </Directory>	
Indexes	если эта опция включена, то при указании <i>URL</i> 'а, ссылающегося на каталог (типа <i>http://www.ru/nirvana</i> или <i>http://localhost/manual</i>), при отсутствии в каталоге файла, описанного директивой DirectoryIndex (<i>index.html, index.php3, index.cgi</i> и т. д.), <i>httpd</i> возвращает клиенту листинг этого каталога
Includes	разрешает выполнение SSI директив в файлах, описанных директивой AddHandler server-parsed <i>mun</i> и находящихся в каталоге <i>/home/httpd/html</i>
FollowSymLinks	позволяет использовать символические ссылки на файлы или каталоги, не находящиеся в <i>/home/httpd/html</i>
AllowOverride None	см. выше
Order allow, deny	определяет последовательность применения правил запрета и правил разрешения доступа к каталогу. Здесь сначала проверяются разрешающие правила, затем запрещающие, т. е. если разрешено, например, по <i>IP</i> , и запрещено, например, по имени <i>хоста</i> , то будет запрещен доступ к этому каталогу, т. к. запрещающие правила сработали последними
Allow from all	разрешен доступ отовсюду.
<Directory /home/httpd/cgi-bin> AllowOverride None Options ExecCGI </Directory>	Прописывает параметры доступа к каталогу <i>/home/httpd/cgi-bin</i> (к тому, где лежат <i>CGI</i> -скрипты). AllowOverride None – см. выше. Options ExecCGI – разрешен запуск <i>CGI</i> и больше ничего
Alias /doc /usr/doc <Directory /usr/doc> order deny,allow deny from all allow from localhost Options Indexes FollowSymLinks </Directory>	Создает <i>алиас</i> <i>/doc</i> для каталога <i>/usr/doc</i> и описывает параметры доступа к нему. order deny,allow – сначала срабатывают запрещающие правила, затем разрешающие. deny from all – запрещен доступ всем клиентам, откуда бы то ни было. allow from localhost – разрешен доступ с машины, на которой и запущен Apache. Options Indexes FollowSymLinks – см. выше. (Если запустить Netscape на той же машине, на которой загружен Apache, и набрать URL – <i>http://localhost/doc</i> или <i>http://127.0.0.1/doc</i> , то можно увидеть листинг каталога <i>/usr/doc</i> .)

Настройка виртуальных серверов в файле httpd.conf

В большинстве случаев один *http*-сервер способен обрабатывать запросы, поступающие на различные, так называемые виртуальные, Web-серверы. Виртуальные серверы могут иметь как один и тот же IP-адрес, но разные доменные имена, так и разные IP-адреса. С точки зрения пользователя второй вариант чуть более предпочтителен, поскольку запрос к серверу, отличающемуся от основного только доменным именем, должен содержать его имя, а некоторые старые браузеры, не поддерживающие протокол HTTP/1.1 (например, Microsoft Internet Explorer 2.0), не включают в запрос эту информацию. Однако такие браузеры выходят из употребления (сейчас их уже менее 0,5 % от общего числа); с другой стороны, выделение собственного IP-адреса каждому виртуальному серверу может быть неоправданной растратой адресного пространства компании.

Для описания адресов и доменных имен виртуальных серверов служат директивы **ServerName**, **ServerAlias**, **NameVirtualHost** и **VirtualHost**. Они необходимы, только если вам нужно установить более одного виртуального сервера.

Директива **ServerName**, находящаяся вне секций **VirtualHost**, определяет имя основного сервера, т. е. сервера, корневой каталог которого задан директивой **DocumentRoot** в файле *srn.conf*. Виртуальные серверы наследуют настройки основного; при необходимости специальной настройки соответствующие директивы помещаются в секции **VirtualHost**, относящейся к данному серверу. Допустимы любые директивы, которые могут встретиться в файлах *httpd.conf* и *srn.conf*, например **DocumentRoot**, **ErrorLog**, **CustomLog**, **Location**, **ServerAdmin**.

Начальная установка и настройка

- I. Распакуйте архив дистрибутива в каталог C:\Apache2.
- II. Откройте в текстовом редакторе файл "C:\Apache2\conf\httpd.conf", являющийся основным конфигурационным файлом сервера Apache.
- III. Правка файла httpd.conf:
 1. Для загрузки модуля `mod_rewrite` найдите и раскомментируйте (уберите в начале строки символ "#") данную строку:

LoadModule rewrite_module modules/mod_rewrite.so

2. Для загрузки PHP интерпретатора, в конец блока загрузки модулей необходимо добавить строку:

LoadModule php5_module "C:/php/php5apache2_2.dll"

3. Определите каталог, содержащий конфигурационный файл PHP, добавив ниже следующую строку:

PHPIniDir "C:/php"

4. Найдите и раскомментируйте строку:

ServerName www.example.com:80

Отредактируйте ее следующим образом, установив изначальное имя сервера:

ServerName localhost:80

5. Найдите строку:


```
DocumentRoot "c:/Apache2/htdocs"
```

Назначьте корневую директорию управления сайтами (немного позже мы ее создадим):

```
DocumentRoot "C:/apache"
```

6. Найдите данный блок:

```
<Directory />  
Options FollowSymLinks  
AllowOverride None  
Order deny,allow  
Deny from all  
</Directory>
```

И замените его на нижеследующий:

```
<Directory />  
Options Includes Indexes FollowSymLinks  
AllowOverride All  
Allow from all  
</Directory>
```

7. Удалите или закомментируйте первоначальный блок управления директориями (он нам не понадобится), который без комментариев выглядит примерно так:

```
<Directory "c:/Apache2/htdocs">  
Options Indexes FollowSymLinks  
AllowOverride None  
Order allow,deny  
Allow from all  
</Directory>
```

8. Найдите блок:

```
<IfModule dir_module>  
DirectoryIndex index.html  
</IfModule>  
Замените его на:  
<IfModule dir_module>  
DirectoryIndex index.html index.htm index.shtml index.php  
</IfModule>
```

9. Найдите строку:

```
ErrorLog "logs/error.log"
```

Замените на нижеследующую (в этом случае просматривать глобальный файл ошибок сервера будет удобнее):

```
ErrorLog "C:/apache/error.log"
```

10. Найдите строку:

```
CustomLog "logs/access.log" common
```

Замените на:

```
CustomLog "C:/apache/access.log" common
```

11. Для работы SSI (включения на стороне сервера) следующие строки, находящийся в блоке <IfModule mime_module>, необходимо найти и раскомментировать:

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

12. Добавьте ниже, в тот же блок <IfModule mime_module>, две строки:

```
AddType application/x-httpd-php .php  
AddType application/x-httpd-php-source .phps
```

13. И, наконец, найдите и раскомментируйте строки:

```
Include conf/extra/httpd-mpm.conf  
Include conf/extra/httpd-autoindex.conf  
Include conf/extra/httpd-vhosts.conf  
Include conf/extra/httpd-manual.conf  
Include conf/extra/httpd-default.conf
```

IV. Сохраните изменения и закройте файл "httpd.conf".

Теперь откройте файл "C:\Apache2\conf\extra\httpd-vhosts.conf" и произведите в нем следующие изменения.

Существующие блоки примеров виртуальных хостов необходимо удалить, и вставить только нижеследующее:

```
NameVirtualHost *:80  
<VirtualHost *:80>  
DocumentRoot "C:/apache/localhost/www"  
ServerName localhost  
ErrorLog "C:/apache/localhost/error.log"  
CustomLog "C:/apache/localhost/access.log" common  
</VirtualHost>
```

V. Сохраните изменения и закройте файл "httpd-vhosts.conf".

Настройка конфигурационных файлов завершена, теперь необходимо установить сервис Apache2.2.

Изменение названия и расширения индексного файла

Обычно индексный файл называется index.htm или index.html. Вы можете по Вашему усмотрению назначить файл с другим именем в качестве индексного файла директории:

DirectoryIndex index.html index.shtml index.pl index.cgi index.php

Индесные файлы будут искаться в указанной последовательности. Если файл не найден, Web-сервер выдаст листинг каталогов и файлов

Ограничение доступа к каталогам без индексного файла

Когда в каталоге отсутствует индексный файл (например, index.html) Web-браузер показывает листинг файлов каталога, позволяя очень любознательным людям видеть внутреннюю структуру Вашего сайта. Так вот, для того, чтобы этого не происходило необходимо включить в файл строку:

options -Indexes

В этом случае при обращении Web-сервер возвращает ошибку 403 (forbidden) - запрещен просмотр директорий.

Ограничение доступа с определенных IP адресов

Иногда бывает необходимо закрыть доступ к сайту для посетителей с определенными IP. В этом случае при попытке доступа к ресурсу посетитель получит сообщение: **Доступ с этого IP для данной ссылки невозможен.**

Это делается в группе настроек <limit method>, где method - способ передачи запроса: GET и\или POST

```
<Limit GET POST>  
Order Allow,Deny  
Deny from 100.99.69.1, 100.99.69.2  
Allow from All  
</Limit>
```

Первая строка - Order Allow,Deny устанавливает приоритет назначения полномочий: сначала разрешения, потом запреты.

Вторая строка - Deny from 100.99.69.1, 100.99.69.2 запрещает доступ к сайту с IP-адресов 100.99.69.1 и 100.99.69.2.

Последняя строка Allow from All разрешает доступ со всех узлов.

Хотя строки и перепутаны местами, но смысл не изменится сначала Allow, затем Deny.

Ограничение доступа к файлам определенного формата

Иногда бывает необходимость закрыть доступ посетителей к файлам с определенного формата (имени расширения). Это делается в группе настроек <Files FileName> - для одного файла и <FilesMatch MASK> - для группы файлов определенного формата, где FileName - имя файла в формате "Имя.Расширение", а MASK - маска имени файла в формате регулярного выражения

```
<FilesMatch "\.(gif/jpeg/jpg/png)$">  
Order Deny,Allow  
Deny from all  
Allow from 100.99.99.1  
</FilesMatch>
```

Первая строка - Order Deny,Allow устанавливает приоритет назначения полномочий: сначала запреты, потом разрешения.

Вторая строка - Deny from all запрещает доступ к файлам с расширениями gif|jpeg|jpg|png со всех узлов.

Последняя строка Allow from 100.99.99.1 разрешает доступ только с IP-адреса 100.99.99.1.

Настройка отображения своих собственных страниц ошибок

Ничто в этом мире непостоянно: файлы могут теряться, ссылки устаревать, а Web-сервера - глючить. Чтобы посетитель видел, что попал именно на Ваш сайт, и у него нет проблем с Web-браузером, рекомендуется настраивать собственные страницы ошибок. Делается это довольно просто:

ErrorDocument 404 <http://www.mysite.net/400.html>

Где 404 - это код ошибки, возвращаемой Web-сервером (подробнее об ошибках можно почитать в статье [Коды ошибок Web-сервера](#)), а <http://www.mysite.net/400.html> - путь к собственной страничке ошибок или редирект на главную страницу сайта.

2.2 Администрирование и установка WEB-СЕРВЕРА

Несколько слов о веб-сервере Apache

Мы используем Apache httpd в качестве основного веб-сервера. Apache используется для организации большинства веб-серверов в мире и является самым массовым продуктом своего класса. Этот сервер обладает обширными возможностями конфигурации, является очень производительным и поддерживает все известные протоколы для работы веб-серверов. Специально для Apache созданы версии таких популярных языков программирования как Perl и PHP, а также этот сервер легко интегрируется с широко применяемыми СУБД (например, MySQL).

Главный сайт проекта находится по адресу <http://apache.org>, а основная документация по версии 1.3.xx доступна на странице <http://apache.org/docs/>.

Пользователям мы предоставляем возможность самостоятельной конфигурации Apache путем использования соответствующих директив в файле [.htaccess](#). Таким образом можно решить большинство задач по конфигурации веб-сервера в условиях массового хостинга.

Индексный файл

Индексный файл или файл-индекс — это тот файл, который открывается по умолчанию при обращении пользователя через веб к каталогу, а не к конкретному файлу. Например, ваш посетитель запросит адрес **http://ваш_домен/price/**, где price — название каталога. Индексный файл это тот файл, который будет показан пользователю при обращении к каталогу без указания имени конкретного файла в нем.

По умолчанию индексными файлами являются следующие: index.html, index.htm, index.php, index.php3, index.phtml, index.shtml, default.htm или default.html. Если вы хотите чтобы первым открывался какой-то иной файл, нужно переопределить текущие значения.

Назначение и использование файла .htaccess

Файл **.htaccess** (обратите внимание, что первый символ в названии файла — точка) применяется для управления веб-сервером Apache со стороны конечного пользователя хостинга. Вы помещаете в этот файл **директивы**, которые веб-сервер воспринимает и обрабатывает, выполняя действия в соответствии с настройками, которые были сделаны пользователем.

Файл .htaccess может быть размещен в корневом каталоге веб-сервера (прямо в каталоге **www**). В этом случае директивы из такого .htaccess действуют по всему веб-серверу. Также .htaccess может находиться и в конкретном подкаталоге сервера. Тогда директивы, которые указаны в этом файле, «перекрывают» действие директив из «основного» файла, который размещен в каталоге **www** или в любом каталоге более высокого уровня. То есть, действие директив из .htaccess наследуется сверху вниз, но не наоборот. Изменения, внесенные в файл, вступают в силу немедленно. Это связано с тем, что информация из .htaccess перечитывается при каждом обращении к веб-серверу Apache.

В .htaccess может быть помещено большинство из **доступных директив** для веб-сервера. Следует заметить, что директивы, в описании которых в поле Context отсутствует упоминание .htaccess недоступны для использования в этом файле конфигурации. На примере директивы AddType видим, что поле Context содержит упоминание о .htaccess, соответственно вы можете ее использовать:

AddType directive

Syntax: AddType *MIME-type extension* [*extension*] ...
Context: server config, virtual host, directory, .htaccess
Override: FileInfo
Status: Base
Module: mod_mime

Пример: как переопределить кодировку html-документов

Мы хотим «объяснить» веб-серверу что все html-документы, которые размещены на сервере, нужно «отдавать» клиенту в кодировке koi8-r, а не в windows-1251, как это сервер делает по умолчанию. Для этого поместим в .htaccess строку:

```
AddType "text/html; charset=koi8-r" .html .htm .shtml
```

Получив такой .htaccess, веб-сервер Apache станет выдавать клиентскому браузеру заголовков, в котором будет указано, что документ имеет кодировку koi8-r.

Если на вашем ресурсе существуют html-документы в разных кодировках, (ISO-8859-1, Windows-1250, Windows-1252, UTF-8), то вам, возможно, будет необходимо отключить принудительную выдачу заголовка с кодировкой windows-1251. Для этого в .htaccess добавляется строка:

```
AddDefaultCharset Off
```

При этом соответствующая кодировка должна быть прописана на каждой html-странице в виде тега `<http-equiv="Content-type" content="text/html; charset=windows-1251" />`

Это примеры простейшего использования возможностей конфигурирования Apache через файл .htaccess.

Пример: как закрыть директорию паролем

Одна из стандартных задач, которая решается путем использования `.htaccess`, это ограничение доступа к определенному каталогу на сервере. Например, нужно дать доступ к определенному каталогу отдельным посетителям, снабдив их при этом уникальным логином и паролем.

В каталоге, к которому хотим ограничить доступ по паролю, создаем файл `.htaccess` с такими директивами:

```
AuthType Basic
AuthName "Some Name"
AuthUserFile /home/uXXXXXX/.htpasswd
require valid-user
```

Путь `/home/uXXXXXX/.htpasswd` обозначает полный путь к файлу паролей на диске нашего сервера. Если, например, вы поместите файл `.htpasswd` (в нем будут пароли) в домашний каталог, куда вы попадаете, зайдя на сервер по FTP, то путь к этому файлу будет иметь вид `/home/uXXXXXX/.htpasswd`, где `uXXXXXX` — наименование вашей виртуальной площадки (например, `u12345`).

В директиве `AuthUserFile` указываем абсолютный путь к файлу с логинами/паролями, который мы создадим чуть позже. Если вы создаете файл `.htaccess` на своем компьютере, а не сразу на сервере при помощи текстового редактора, обратите внимание на то, что `.htaccess` должен передаваться по FTP **строго в текстовом (ASCII) режиме**.

Создаем файл паролей. Файл с паролями должен содержать строки вида **login:password**. Пароль должен быть зашифрован с использованием алгоритма MD5. Один из способов создать такой файл — воспользоваться программой, входящей в поставку Apache — `htpasswd` (возможный путь — `/usr/local/bin/htpasswd`).

Рассмотрим, как создать файл паролей в **unix shell** прямо на сервере. Зайдем в `shell`, и будем выполнять следующие команды:

- `htpasswd -mbc .htpasswd user1 sNQ7j9oR2w`

создаем новый файл `.htpasswd`, в который добавляем запись для пользователя `user1` с паролем, указанным в командной строке. Просьба **обязательно** заменить `sNQ7j9oR2w` на любой собственный пароль — здесь этот пароль указан только для примера

- `htpasswd .htpasswd user2`

добавляем в уже существующий файл `.htpasswd` пользователя `user2`, а пароль вводим вручную в ответ на соответствующий запрос программы

Если вы используете Windows и не хотите пользоваться `unix shell` для генерации паролей, можно загрузить Windows-версию программы `htpasswd` [здесь](#) и создать файл с паролями на своем компьютере, после чего загрузить его на сервер. Если у вас уже установлена Windows-версия Apache, файл **htpasswd.exe** можно найти в каталоге **Program Files\Apache Group\Apache\bin**.

Итак, получите `htpasswd.exe` и используйте его для генерации паролей таким образом:

- `htpasswd.exe -mc .htpasswd user1`

создаем новый файл паролей `htpasswd.exe`, пароль и его подтверждение будут запрошены интерактивно

- `htpasswd.exe -m .htpasswd user2`

добавляем пользователя `user2` в существующий файл паролей `htpasswd.exe`, запросив пароль интерактивно

После окончания заведения всех логинов файл нужно загрузить на сервер.

Пример: переопределение индексного файла

Ситуация: пользователь обратился к каталогу http://www.ваш_домен.ru/price/. При таком запросе первым откроется и будет показан индексный файл. Если вы хотите переопределить индексный файл и сделать так, чтобы первым открывался не `index.htm`, а, например, файл `myindex.php`, то сделать это можно поместив в файл `.htaccess` в соответствующем каталоге следующую инструкцию:

DirectoryIndex myindex.php

Получив `.htaccess` с таким содержимым, веб-сервер Apache откроет по умолчанию именно файл **myindex.php**.

Пример: запрет и разрешение выдачи листинга

В ряде случаев требуется выводить список файлов в каталоге (листинг каталога) в случае отсутствия в каталоге файла, который показывается по умолчанию. Для этого необходимо добавить в `.htaccess` следующую строку:

Options +Indexes

Файл `.htaccess` необходимо создавать именно в том каталоге, в котором планируется разрешить листинг. Данная директива будет действовать также и на все подкаталоги (это достигается включенной по умолчанию в настройках виртуального хоста директивой **AllowOverride All**).

По умолчанию включена директива **Options-Indexes**, и в случае отсутствия индексной страницы вы получите HTTP ошибку 403.

Пример: собственные страницы ошибок

Иногда посетители веб-сервера запрашивают страницы, которые по каким-то причинам на сервере не существуют: неправильная ссылка с другой страницы или с другого сайта, владелец сервера случайно удалил документ и так далее. По умолчанию Apache выдает некую довольно аскетичную страницу, на которой находится сообщение вроде «File not found». Вы можете создать альтернативную версию этой страницы, задав обработчик этой ошибки через `.htaccess`.

Пример: запрет доступа с некоторых IP-адресов

Иногда возникает необходимость запретить доступ к сайту или его части с некоторых IP-адресов.

В таком случае необходимо создать в нужной директории файл `.htaccess` с директивами. Например, чтобы запретить доступ с IP-адреса 172.16.16.16:

*Order Allow,Deny
Allow from All
Deny from 172.16.16.16*

Теперь при попытке обратиться к сайту с IP-адреса **172.16.16.16** посетитель получит ошибку 403 или вашу страницу для этой ошибки.

Указание части адреса в виде **172.16.16** ограничит доступ из подсети **172.16.16/24**.

С более подробной документацией вы можете ознакомиться в [документации по Apache](#).

Пример: запрет доступа к некоторым файлам

Иногда возникает необходимость запретить доступ к определенным файлам. Например, к конфигурационным файлам, содержащим реквизиты доступа к базам данных, интерфейсам и т.п. Допустим, в файле **config.cfg** вы храните логин/пароль доступа к базе данных. Создаем в этой директории файл **.htaccess** с директивами:

```
<FilesMatch "\.(cfg)$">  
Order allow,deny  
Deny from all  
</FilesMatch>
```

Теперь, если посетитель наберет в браузере нечто вида **http://www.ваш_домен.ru//config.cfg**, он получит ошибку 403 или вашу страницу для этой ошибки.

Пример: заголовок last-modified

В ряде случаев требуется, чтобы web-сервер выдавал HTTP-заголовок **Last-Modified**. К примеру, при регистрации вашего ресурса на [Яндексе](#), возникает ошибка «**Неправильные даты**». Для статических документов сервер будет выдавать значение last-modified всегда. Это действительно для html-файлов. Для **SSI** сервер будет выдавать значение last-modified в том случае, если прописана директива «XBitHack full» (просто пропишите эту строку в **.htaccess**), и для файла, к которому происходит обращение, выставлен атрибут «исполняемый» для группы. В скриптах last-modified выдается иными средствами. Например, если учесть то, что php-скрипт генерирует код динамически, то самым логичным будет в качестве last-modified отдавать текущую дату и время./>

Реализуется это следующим образом:

```
<? header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); ?>
```

Внимание: команда **header** должна выполняться в php-скрипте до того, как скрипт начнет выдавать html-текст в браузер пользователя.

Полезные ссылки по теме:

- [Функция header](#)
- [Функции для работы с сервером Apache](#)

Пример: управление кэшированием

Для того, чтобы сайт работал максимально эффективно, целесообразно устанавливать время кэширования для различных типов файлов на максимально возможный срок. Для этого существует модуль Apache **mod_expires**.

Настройка параметров модуля **mod_expires** производится в файле **.htaccess**, что позволяет сделать индивидуальные настройки для каждого каталога.

В приведенном ниже примере отключено кэширование для текстовых документов, установлен период обновления для файлов с расширением gif — 3 месяца с момента изменения файла, для файлов с расширением jpeg — 1 день с момента обращения:

```
ExpiresActive on  
ExpiresByType image/jpeg "access plus 1 day"  
ExpiresByType image/gif "modification plus 3 months"  
ExpiresByType text/html "now"
```


*Примечание: При использовании **Parser** если вам необходимо полностью исключить заголовки от модуля **Apache mod_expires**, то для этого необходимо в директории с **parser3.cgi** создать файл **.htaccess** и внести в него следующую директиву: **ExpiresActive off***

Пример: как создать переадресацию

- Если у вас размещены 2 домена (не обязательно на одной площадке) **domain1.tld** и **domain2.tld**, и вам необходимо, чтобы при обращении к **domain2.tld** у пользователей изменялся адрес на «правильный», и сразу происходило перенаправление, тогда добавьте для домена **domain2.tld** переадресацию на **http://domain1.tld/**. О том, как это сделать, написано в следующей статье.
- Если вам необходимо, чтобы при обращении к вашему домену **domain.tld** происходило автоматическое перенаправление на **www.domain.tld**, создайте на виртуальной площадке в директории **/home/uXXXX/domain.tld/www** файл **.htaccess** (обратите внимание на то, что название файла начинается с точки) следующего содержания:
- RewriteEngine on
- RewriteCond %{HTTP_HOST} ^domain\.tld

```
RewriteRule ^(.*)$ http://www.domain.tld/$1 [R=permanent,L]
```

где **uXXXX** — имя вашей виртуальной площадки, **domain.tld** — имя вашего домена.

- О создании переадресаций с другими условиями, вы можете узнать из документации по веб-серверу **Apache**.

Переадресация с HTTP на HTTPS при использовании IP+SSL

Если к домену подключена услуга Выделенный IP+SSL и вы хотите запретить HTTP, оставив посетителям доступ только по защищенному протоколу HTTPS, добавьте в **.htaccess** следующее правило:

```
RewriteEngine On
```

```
RewriteCond %{HTTP:PORT} !^443$
```

```
RewriteRule ^(.*)$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R]
```

Пример: особенность переадресации на синонимах

Предположим, есть домен **domain1.tld** и синоним **domain2.tld**. Если запросить в браузере адрес **http://domain2.tld/dir/** со знаком слэша в конце, то будет отображена индексная страница из директории **dir** основного домена, при этом содержимое адресной строки браузера останется без изменений. Но если запросить **http://domain2.tld/dir** без слэша в конце, то произойдет переадресация на **http://domain1.tld/dir/**, и содержимое адресной строки изменится соответствующим образом.

Такова особенность работы модуля **mod_dir**, при которой если происходит запрос файла, являющегося директорией, но запрос не оканчивается знаком слэш, то **mod_dir** осуществляет внешнюю переадресацию на тот же адрес со знаком слэша в конце. В случае синонима при переадресации заменяется и имя домена.

Если такое поведение веб-сервера вас не устраивает, добавьте в файл **.htaccess** следующие строки:

```
RewriteEngine on
```

```
RewriteCond %{REQUEST_FILENAME} -d
```

```
RewriteCond %{REQUEST_URI} !^domain2.tld$
```

```
RewriteRule ^(.+[/\])$ http://domain2.tld/$1/ [R]
```

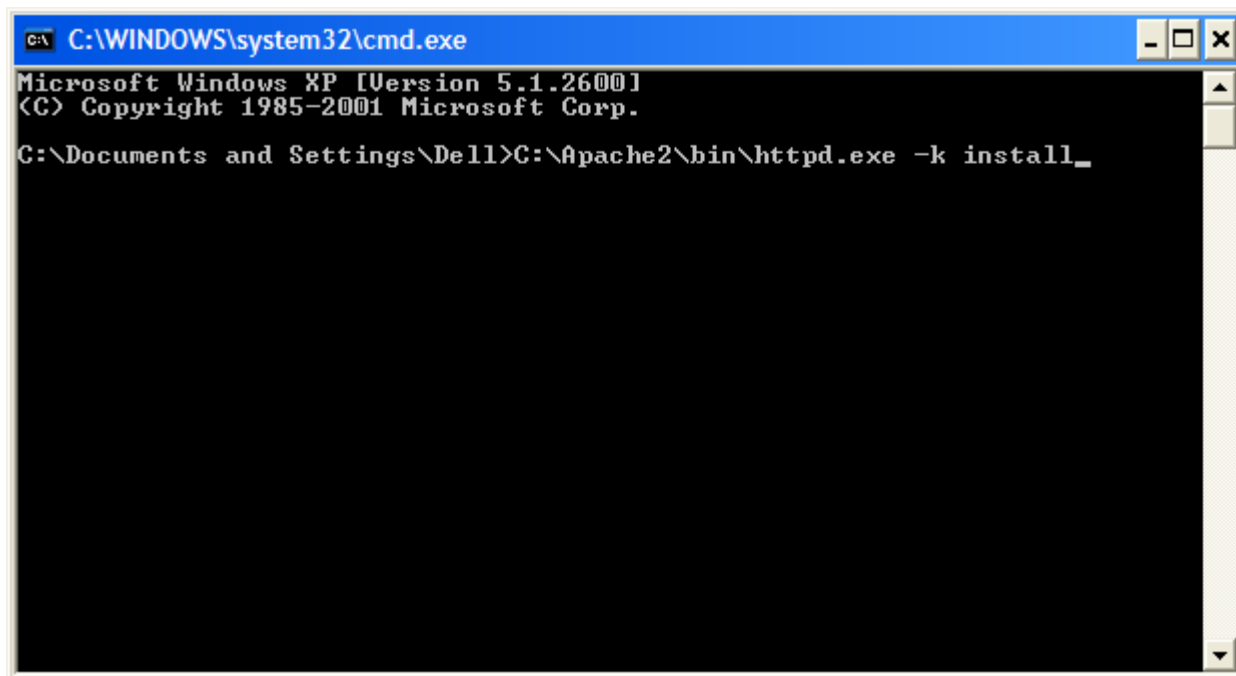
О создании переадресаций с другими условиями вы можете узнать из [документации по web-серверу Apache](#).

Установленные модули Apache

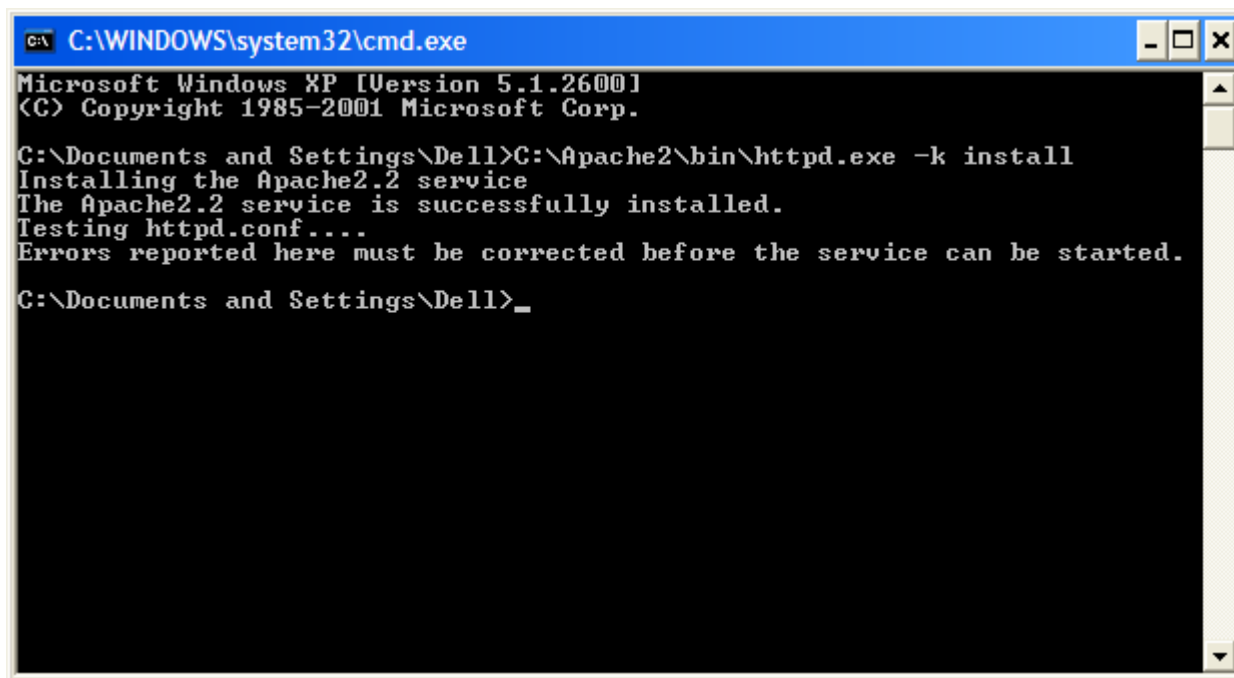
Список стандартных и дополнительных модулей для веб-сервера Apache, которые установлены на хостинговых машинах:

- mod_env
- mod_log_config
- mod_mime
- mod_status
- mod_include
- mod_autoindex
- mod_dir
- mod_cgi
- mod_actions
- mod_alias
- mod_rewrite
- mod_access
- mod_auth
- mod_expires
- mod_setenvif
- mod_php

Установка сервиса Apache2.2 в картинках



Запускаем командную строку и вводим в ней "C:\Apache2\bin\httpd.exe -k install"



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Dell>C:\Apache2\bin\httpd.exe -k install
Installing the Apache2.2 service
The Apache2.2 service is successfully installed.
Testing httpd.conf....
Errors reported here must be corrected before the service can be started.

C:\Documents and Settings\Dell>_
```

Вот что мы должны увидеть при успешной установке сервиса Apache2.2. Если все прошло успешно, движемся дальше - устанавливаем ручной запуск сервиса Apache2.2 для чего проходим путь: "Start" ("Пуск") → "Control Panel" ("Панель управления") → "Administrative Tools" ("Администрирование") → "Services" ("Службы"), в открывшемся окне управления служб выбираем строку "Apache2.2" и делаем на ней двойной щелчок, после чего во вкладке "General" ("Общие") выбираем ручной запуск сервиса - "Startup type: Manual" ("Тип запуска: вручную"). Это необходимо сделать для того, чтобы лишние службы не загружали систему. Учитывая, что домашний компьютер используется не только для веб-разработок, но и для многих других нужд, ручной запуск и остановка, непостоянно используемых сервисов, является наиболее приемлемым.

Создание структуры каталогов виртуальных хостов.

В корне диска необходимо создать каталог "apache" - в нем будут лежать ваши виртуальные хосты (домены), глобальный файл журнала ошибок "error.log" (создается программой при первом запуске, автоматически), глобальный файл доступа "access.log" (создается автоматически). В каталоге "apache" создаем еще одну пустую папку - "localhost", в которой, в свою очередь, создаем папку "www", именно в последней и надо будет держать наше добро в виде локальных скриптов. Такая, казалось бы странная структура каталогов, продиктована схожей схемой построения каталогов в системах Unix, и призвана упростить в дальнейшем ее понимание и использование.

Пример создания виртуального хоста.

При необходимости установки собственных виртуальных хостов сделайте следующее: Откройте файл "httpd-vhosts.conf", и создайте в нём блок, примерно, следующего содержания:
<VirtualHost *:80>

```
# Папка, в которой будет корень вашего хоста.
DocumentRoot "C:/apache/test.ru/www"
# Домен по которому вы сможете обращаться к виртуальному хосту.
ServerName test.ru
# Алиас (добавочное имя) домена.
```

```
ServerAlias www.test.ru  
# Файл, в который будут записываться ошибки.  
ErrorLog "C:/apache/test.ru/error.log"  
# Файл журнала доступа к хосту.  
CustomLog "C:/apache/test.ru/access.log" common  
</VirtualHost>
```

Затем в каталоге "apache", создайте папку "test.ru" (прямо так, с точкой), в которой, в свою очередь, создайте папку "www".

Следующий шаг создания виртуального хоста – это изменение файла C:\WINDOWS\system32\drivers\etc\hosts операционной системы. Откройте данный файл и добавьте в него две строки:

```
127.0.0.1 test.ru  
127.0.0.1 www.test.ru
```

Теперь запустите сервер Apache выполнив в командной строке "C:\Apache2\bin\httpd.exe -k start", откройте браузер, введите в адресной строке "test.ru" или "www.test.ru" и вы окажетесь в своем виртуальном хосте. Только будьте внимательны, теперь вы сможете попасть на оригинальный сайт с именем виртуального хоста ("www.test.ru", если таковой существует), только закомментировав либо удалив строку: "127.0.0.1 www.test.ru", в вышеупомянутом файле "hosts".

Документация Apache, при запущенном сервере, доступна по адресу <http://localhost/manual/>.

Остановить работу Apache можно выполнив в командной строке "C:\Apache2\bin\httpd.exe -k stop". При необходимости перезагрузить Apache, выполните в командной строке "C:\Apache2\bin\httpd.exe -k restart".

Установка и настройка веб-сервера Apache – завершена.

Создание пакетных файлов для запуска и остановки сервисов

Согласитесь, что вручную редактировать файл "hosts" при каждом запуске сервисов – это неудобно, поэтому для более удобного, одновременного запуска сервисов Apache, MySQL, и изменения файла "hosts" мы создадим два пакетных файла: на запуск и остановку, которые будут выполнять всю рутинную работу автоматически.

При использовании виртуальных хостов необходимо создать в директории C:\apache два файла: vhosts-off.txt – содержащий изначальное содержимое файла "hosts" и vhosts-on.txt – содержащий все виртуальные хосты. Обратите внимание, что при создании новых виртуальных хостов вам необходимо будет добавлять их в файл vhosts-on.txt, а не в C:\WINDOWS\system32\drivers\etc\hosts. Посмотрите на примеры ниже.

Файл vhosts-off.txt (может содержать одну единственную строку):

```
127.0.0.1 localhost
```

Пример файла vhosts-on.txt с виртуальными хостами www.test.ru и test.ru:

```
127.0.0.1 localhost  
127.0.0.1 www.test.ru  
127.0.0.1 test.ru
```

В той же директории C:\apache, создайте два пакетных файла: start-webserver.bat – для

запуска сервисов и подмены файла "hosts", и stop-webserver.bat – для остановки сервисов и очистки файла "hosts".

Файл запуска start-webserver.bat:

```
@echo off
echo.
if not exist C:\apache\vhosts-on.txt goto no_vhosts
echo Create virtual hosts:
copy /v /y C:\apache\vhosts-on.txt C:\WINDOWS\system32\drivers\etc\hosts
echo.
:no_vhosts
NET start Apache2.2
NET start MySQL
```

Файл остановки stop-webserver.bat:

```
@echo off
echo.
if not exist C:\apache\vhosts-off.txt goto no_vhosts
echo Restore hosts file:
copy /v /y C:\apache\vhosts-off.txt C:\WINDOWS\system32\drivers\etc\hosts
echo.
:no_vhosts
NET stop Apache2.2
NET stop MySQL
```

В случае если вы не используете виртуальные хосты или хотите запустить сервисы без подмены файла "hosts", просто уберите из директории C:\apache файлы vhosts-on.txt и vhosts-off.txt.

Одновременно с сервисами удобно запускать программу Apache Monitor, отображающую в системном лотке состояние сервера Apache, для чего можно создать еще один пакетный файл "start-webserver-monitor.bat", содержание которого аналогично файлу "start-webserver.bat" с добавлением в самый конец следующей строки:

```
start "" "C:\Apache2\bin\ApacheMonitor.exe"
```

Теперь для запуска всего инструментария вам понадобится запустить файл "start-webserver-monitor.bat", или "start-webserver.bat", а для остановки "stop-webserver.bat". При желании вы можете переименовать эти файлы, переместить их в любое другое место из папки "C:\apache", либо создать на данные файлы ярлыки, например, на рабочий стол.

Замечания по Linux

Для редактирования файлов конфигурации и навигации по файловой системе удобно использовать программу **mc**, для ее загрузки необходимо набрать в командной строке:

```
[root@lis] $ mc
```

Интерфейс программы интуитивно понятен и не представляет трудностей при использовании.

Файлы конфигурации web-сервера **Apache** в данной системе находятся в каталоге:

```
/etc/httpd/conf/
```

Для запуска, остановки, перезапуска web-сервера используются следующие скрипты (программы):

Остановка: /etc/rc.d/init.d/httpd stop.

Запуск: /etc/rc.d/init.d/httpd start.

Перезапуск: /etc/rc.d/init.d/httpd restart.

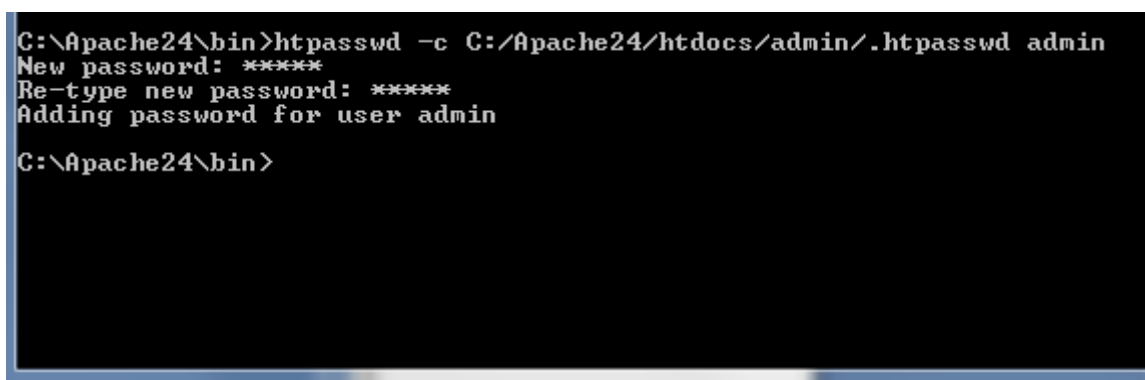
Пример выполнения лабораторной работы

Вариант 0:

Настройки VirtualHost: Создать папку admin в корне сайта и ограничить к ней доступ при помощи basic авторизации, используя файл .htaccess. Реализовать CGI скрипт — вывод содержимого указанного каталога.

Выполнение работы:

Этап первый — создание файла с именами пользователей и их зашифрованными паролями. Используется утилита htpassw (рисунок 1). Полученный в результате файл представлен на рисунке 2.



```
C:\Apache24\bin>htpasswd -c C:/Apache24/htdocs/admin/.htpasswd admin
New password: *****
Re-type new password: *****
Adding password for user admin
C:\Apache24\bin>
```

Рис 1. Работа утилиты htpassw

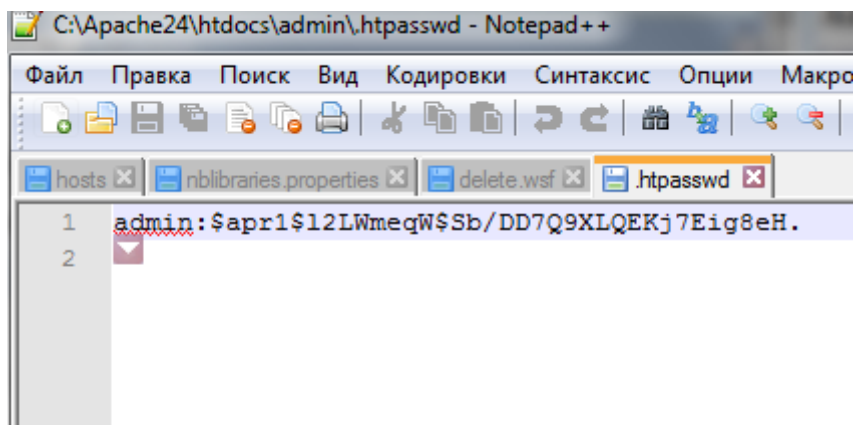


Рис 2. Полученный файл с пользователями

Второй этап — создание конфигурации. На рисунке 3 представлен конфигурационный файл.

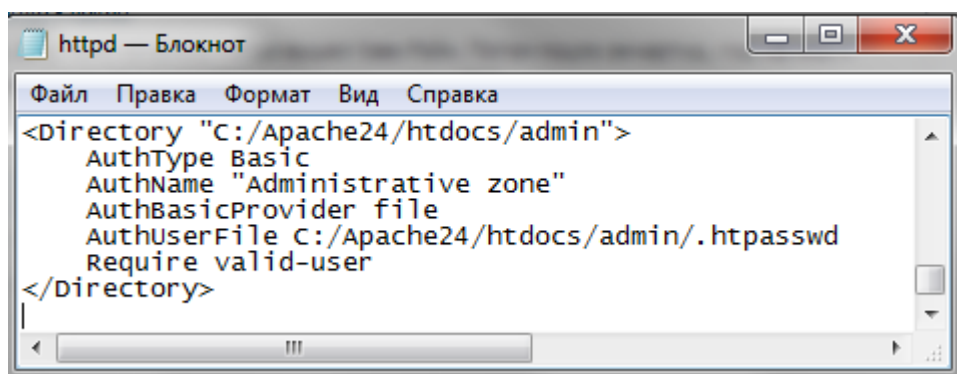


Рис 3. Изменения в конфигурационном файле

Проверка работоспособности представлена на рисунках 4, 5 и 6.

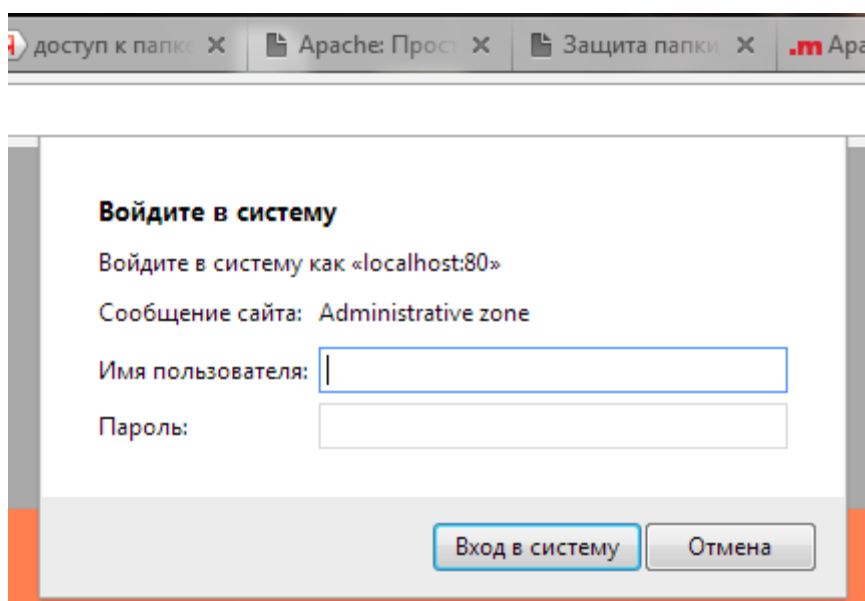


Рис 4. Аутентификация

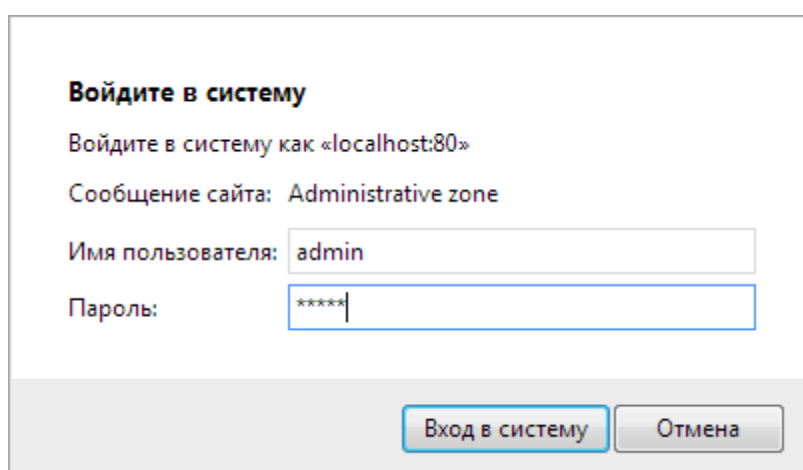


Рис 5. Аутентификация

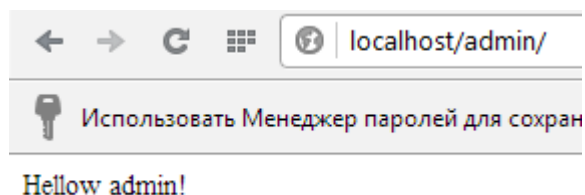


Рис 6. Аутентификация

CGI скрипт представлен на рисунке 7.

```
hosts x nblibraries.properties x delete.wsf x .htpasswd x printenv.pl x
1  #!C:/Perl64/bin/perl
2  #
3
4  # To permit this cgi, replace # on the first line above with the
5  # appropriate #!/path/to/perl shebang, and on Unix / Linux also
6  # set this script executable with chmod 755.
7  #
8  # ***** !!! WARNING !!! *****
9  # This script echoes the server environment variables and therefore
10 # leaks information - so NEVER use it in a live server environment!
11 # It is provided only for testing purpose.
12 # Also note that it is subject to cross site scripting attacks on
13 # MS IE and any other browser which fails to honor RFC2616.
14
15 ##
16 ##  printenv -- demo CGI program which just prints its environment
17 ##
18 #use strict;
19 #use warnings;
20
21 #print "Content-type: text/plain; charset=iso-8859-1\n\n";
22 #foreach my $var (sort(keys(%ENV))) {
23 #   my $val = $ENV{$var};
24 #   $val =~ s|\n|\\n|g;
25 #   $val =~ s|"|\\\"|g;
26 #   print "$var=\"$val\"\n";
27 #}
28 print "Content-Type: text/html\n\n";
29 print "<html> \n";
30 opendir(Dir, $INC[2]) || die;
31
32 while ($file = readdir(Dir))
33 {
34     print "$file \n"
35 }
36
37 closedir(Dir);
38 print "</html> \n";
```

Рис 7. Скрипт

Демонстрация работы скрипта приведена на рисунках 8 и 9.

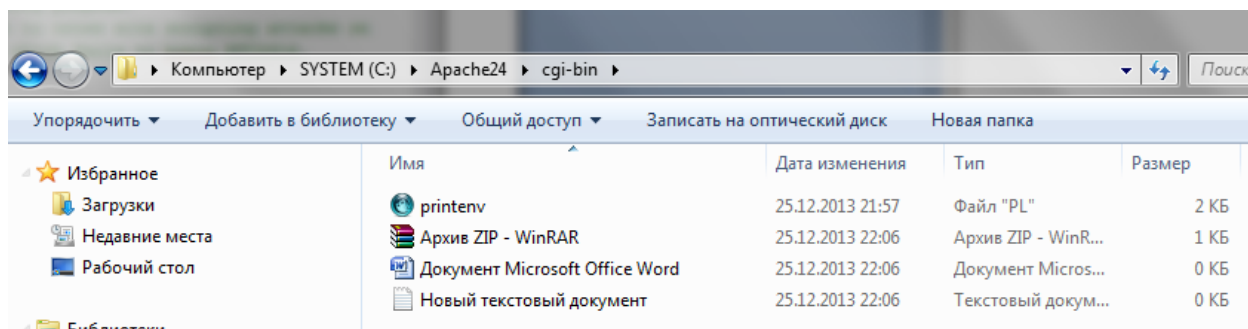


Рис 8. Каталог с файлами

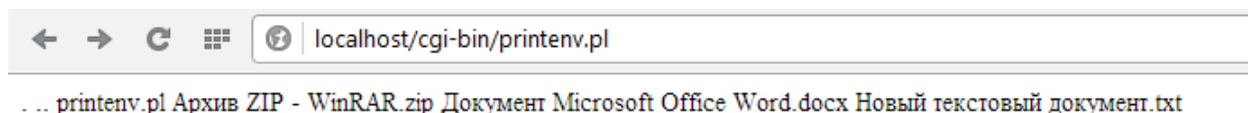


Рис 9. Вывод содержимого каталога

3. Задания для выполнения

В ходе лабораторной работы необходимо произвести настройку web-сервера в соответствии с заданием, продемонстрировать его работоспособность преподавателю.

Для демонстрации работоспособности необходимо переписать с <ftp://ais.khstu.ru/incoming> средствами Linux на локальный компьютер каталоги testserver1 и testserver2, для чего использовать в программе **mc** пункт меню **left (right) -> ftp link**. В данных каталогах содержатся тестовые сайты:

- проверка исполнения файла /cgi-bin/test1.cgi
- проверка исполнения файла /cgi-bin/test2.pl
- проверка исполнения файла /cgi-bin/test3
- проверка SSI-включения файла /cgi-bin/test1.cgi в файл test4.html
- проверка SSI-включения файла /cgi-bin/test1.cgi в файл test4.shtml

Для создания виртуальных серверов по доменным именам используемые доменные имена необходимо прописать в файле c:\winnt\system32\driver\etc\hosts на том компьютере, на котором будет проводиться проверка.

При защите необходимо знать назначение используемых директив, уметь объяснить информацию, содержащуюся в log файлах.

4. Варианты индивидуальных заданий.

1. Настройки VirtualHost: Ограничить доступ к сайту по IP адресу: запретить доступ с IP 172.12.12.13. Реализовать CGI скрипт — проверки логина и пароля.
2. Настройки VirtualHost: Ограничить доступ по доменному имени — запретить переходы из бесплатных каталогов catalog.ru и freecatalog.org. Реализовать CGI скрипт — вывода переменных окружения.
3. Настройки VirtualHost: Запретить скачивать файлы-мультимедиа (mp3, avi). Реализовать CGI скрипт — вывод таблицы переданных параметров методом GET.
4. Настройки VirtualHost: Закрыть доступ к сайту в ночное время. Реализовать CGI скрипт — вывод таблицы переданных параметров методом POST.
5. Настройки VirtualHost: Расширить список индексных файлов: + index.htm +index.php. Реализовать CGI скрипт - вывод информации о пользователе (IP адрес, Браузер, ОС).
6. Настройки VirtualHost: Запретить вывод содержимого каталога при отсутствии индексного файла. Реализовать CGI скрипт - вывод текущей даты и времени.

7. Настройки VirtualHost: Установить кодировку UTF-8. Реализовать CGI скрипт — вывод информации о настройках сетевых интерфейсов.
8. Настройки VirtualHost: включить максимальный уровень журналирования. Реализовать CGI скрипт — вывод информации о свободном месте на дисках.
9. Настройки VirtualHost: включить распознавание файлов 7z, как архивных. Реализовать CGI скрипт — вывод информации о размере файла index.html.
10. Настройки VirtualHost: переключить сервер на работу на порту 8080. Реализовать CGI скрипт – вывод количества запущенных процессов httpd.
11. Настройки VirtualHost: создать доступ к сайту по паролям и ролям. Простым пользователям к сайту, администраторам к папке admin. Реализовать CGI скрипт – вывод владельцев файлов в каталоге.

5. Контрольные вопросы.

1. Какие файлы содержат конфигурационную информацию web-сервера?
2. Какова последовательность установки web-сервера?
3. Как проверить работоспособность web-сервера?
4. Где хранятся log файлы?
5. Что такое виртуальный web-сервер?

6. Список рекомендуемой литературы.

1. Бэндел Дэвид. Защита и безопасность в сетях Linux = Linux security toolkit: Пер. с англ. / Бэндел Дэвид. - СПб.: Питер, 2002. – 480 с.
2. Блэк У. Интернет: Протоколы безопасности = Internet Security Protocols.Protecting IP Traffic: Пер. с англ. / У. Блэк. – СПб.: Питер, 2001. – 288 с.
3. Попов В. Б. Основы компьютерных технологий: Учеб. пособие для вузов / В. Б. Попов. – М.: Финансы и статистика, 2002. - 704с.
4. Уэйнгроу К. UNIX. Администрирование. – М.: ДМК Пресс, 2002. – 416 с.
5. Холден Г. Apache Server в комментариях. – М.: DiaSoft, 2000. – 480 с.
6. Рич Б. Apache. Настольная книга администратора. – М.: DiaSoft, 2002. –378 с.
7. Хокинс С. Администрирование Web-сервера Apache и руководство по электронной коммерции. – М.: Вильямс, 2001. – 330 с.
8. Ньюкомер Э. Веб-сервисы. XML, WSDL, SOAP и UDDI. – СПб.: Питер, 2003. – 256 с.