ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

АДМИНИСТРИРОВАНИЕ LINUX-СЕРВЕРОВ

Методические рекомендации к лабораторным работам для студентов направления подготовки 09.03.04 «Программная инженерия» дневной формы обучения

Могилев 2018

УДК 004.4
ББК 32.973.202
А 31

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «13» марта 2018 г., протокол № 11

Составитель канд. техн. наук, доц. И. А. Евсеенко

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации предназначены для студентов направления подготовки 09.03.04 «Программная инженерия» дневной формы обучения.

Учебно-методическое издание

АДМИНИСТРИРОВАНИЕ LINUX-СЕРВЕРОВ

Ответственный за выпуск	А. И. Якимов
Технический редактор	С. Н. Красовская
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Основной материал в приведенных методических рекомендациях посвящен серверным операционным системам семейства Linux.

Целью методических рекомендаций является предоставление возможности самостоятельного освоения студентами основных методов администрирования локальных сетей на основе серверных операционных систем Linux, получения практических навыков администрирования, а также изучения перспективных направлений в развитии современных серверных операционных систем Linux.

Целью дисциплины является изучение проблем администрирования Linux серверов, классификации операционных систем и применяемого программного обеспечения для администрирования серверов Linux, а также теоретических и практических основ администрирования локальных сетей на основе операционных систем Linux.

При освоении данной дисциплины студент приобретает практические навыки работы с серверными и клиентскими опрерационными системами Linux, создания клиент-серверных приложений, выбора и настройки операционных систем и программного обеспечения для них, диагностирования и устранения неполадок в локальных сетях, инсталлирования, тестирования и настройки конкретных конфигураций операционных систем, работы с различными операционными системами и их администрирования, а также настройка удаленного доступа к серверу и Интернету подключения.

1 Лабораторная работа № 1. Установка операционной системы Linux. Основные принципы функционирования ОС Linux

Цель работы: изучение архитектуры и принципов функционирования многопользовательской многозадачной операционной системы Linux, особенности ее использования в качестве сервера и рабочей станции.

Система Linux включает следующие основные компоненты.

Ядро. Выполняет функции управления памятью, процессорами. Осуществляет диспетчеризацию выполнения всех программ и обслуживание внешних устройств. Все действия, связанные с вводом/выводом и выполнением системных операций, выполняются с помощью системных вызовов. Системные вызовы реализуют программный интерфейс между программами и ядром. Имеется возможность динамического конфигурирования ядра.

Диспетчер процессов Init. Активизирует процессы, необходимые для нормальной работы системы и производит их начальную инициализацию. Обеспечивает завершение работы системы, организует сеансы работы пользователей, в том числе для удаленных терминалов.

Интерпретатор команд Shell. Анализирует команды, вводимые с терминала либо из командного файла, и передает их для выполнения в ядро системы. Команды обычно имеют аргументы и параметры, которые обеспечивают модернизацию выполняемых действий. Shell является также языком программирования, на котором можно создавать командные файлы (shell-файлы). При входе в ОС пользователь получает копию интерпретатора shell в качестве родительского процесса. Далее, после ввода команды пользователем создается порожденный процесс, называемый процессом-потомком. То есть после запуска ОС каждый новый процесс функционирует только как процесс-потомок уже существующего процесса. В ОС Linux имеется возможность динамического порождения и управления процессами.

Shell-интерпретатор в соответствии с требованиями стандарта POSIX поддерживает графический экранный интерфейс, реализованный средствами языка программирования Tcl/Tk.

Обязательным в системе является интерпретатор Bash, полностью соответствующий стандарту POSIX. В качестве Shell может быть использована оболочка mc с интерфейсом, подобным Norton Commander.

Сетевой графический интерфейс X-сервер (X-Windows). Обеспечивает поддержку графических оболочек.

Графические оболочки KDE, Gnome. Отличительными свойствами KDE являются: минимальные требования к аппаратуре, высокая надежность, интернационализация. Базовые библиотеки KDE (qt, kde-libs) признаны одними из лучших продуктов по созданию графического интерфейса, обеспечивают простое написание программ с использованием передовых технологий. Gnome имеет развитые графические возможности, но более требователен к аппаратным средствам.

Сетевая поддержка NFS, SMB, TCP/IP. NFS – программный комплекс PC-NFS (Network File System) для выполнения сетевых функций. PC-NFS ориентирован для конкретной ОС персонального компьютера (PC) и включает драйверы для работы в сети и дополнительные утилиты. SMB – сетевая файловая система, совместимая с Windows NT. TCP/IP – протокол контроля передачи данных (Transfer Control Protocol/Internet Protocol). Сеть по протоколам TCP/IP является неотъемлемой частью ОС семейства UNIX. Поддерживаются любые сети, от локальных до Интернета, с использованием только встроенных сетевых средств.

Инструментальные средства программирования. Основой средств программирования является компилятор GCC или его экспериментальные версии EGCS и PGCC для языка C++; модули поддержки других языков программирования (Obective C, Java и др.); интегрированные среды и средства визуального проектирования: Kdevelop, Xwpe; средства адаптации привязки программ AUTOCONFIG, AUTOMAKE.

Задание

1 Загрузите виртуальную машину, используя указанный преподавателем образ, и выполните настройку компонентов сетевых подключений для работы в сети.

2 Запустите установку операционной системы Linux.

3 Создайте разметку жесткого диска и выберите раздел для установки.

4 Настройте время и дату.

5 Установите поддержку сети.

6 Настройте Главное меню, регистрацию компонентов и сохраните настройки.

7 Установите драйверы устройств. В процессе установки могут потребоваться драйверы для устройств, для которых в БД драйверов системы нет соответствующего драйвера. Если вы занимаетесь в группе под руководством преподавателя, то он предоставит все необходимые драйверы. Если вы занимаетесь индивидуально, то сами позаботьтесь о необходимых драйверах для ваших компьютеров.

Контрольные вопросы

1 Перечислите основные функции и назначение многопользовательской многозадачной операционной системы Linux и ее отличительные особенности от однопрограммной системы DOS.

2 Какое назначение имеет ядро системы и интерпретатор команд?

3 В чем заключается понятие «процесс» и какие операции можно выполнить над процессами?

4 Как задаются и выполняются простые и сложные команды?

5 Какие функции выполняет командный интерпретатор Shell?

2 Лабораторная работа № 2. Терминал и командная оболочка операционной системы Linux

Цель работы: приобрести опыт работы с командной строкой ОС Linux, изучить основные команды (рабочая станция, рабочая директория, пользователи, дата, календарь, список процессов, завершение работы), освоить навыки работы в терминале Linux, научиться создавать новых пользователей при помощи терминала Linux.

Стандартные команды в Linux отличаются от команд DOS и Windows – обычно они короче. При работе с командной строкой как обычно мигающий курсор обозначает позицию ввода текста, командная строка начинается с текущего пути и имени компьютера, за которым следует символ $, % или #. Последний означает, что команды будут выполняться от имени суперпользователя root. Символ ~ означает путь к текущей домашней директории пользователя.

Большинство команд в Linux, не требующих вывода информации пользователю, в случае успешного завершения вообще ничего не выводят на экран. Выводятся только ошибки и предупреждения в случае нарушения нормального выполнения команды. То есть в Linux действует общий принцип «молчит, значит работает».

В любом терминале Linux стрелками вверх/вниз на клавиатуре можно листать историю команд, которая сохраняется между сеансами работы и различается для разных пользователей и хостов. Набранное частично команда или имя файла или каталога в текущей директории может быть автоматически дописано клавишей TAB. Если найдено более одного варианта и однозначно продолжить команду по TAB невозможно, то выводятся все подходящие варианты.

При работе в графической среде удобны эмуляторы терминала. Как правило они поддерживают закладки – несколько терминалов в одном окне, поддерживают цветовые схемы. Наиболее распространены эмуляторы терминала Gnome Terminal, Konsole, XFCE Terminal.

Терминал – эмулятор консоли. Именно в терминале мы будем работать с CLI (интерфейсом командной строки). Терминал часто также называют консолью или шеллом (от англ. «shell» – оболочка).

Многие пользователи и в особенности администраторы серверов под Linux в работе используют именно консоль, а не графическую оболочку, это связано с тем, что настройка и конфигурация Linux в основном заключается в редактировании текстовых конфигурационных файлов. Даже если вы являетесь простым пользователем ОС Linux, большинство инструкций по настройке написаны с использованием консоли и необходимо знать основные команды.

Стоит обратить внимание на системные каталоги ОС, в которых находятся файлы, необходимые для управления и сопровождения системы, а также стандартные программы. Их имена, расположение и содержание одинаковы почти во всех ОС Linux, поэтому эти каталоги называют также стандартными. Впрочем, на данный момент эпитет «стандартные» отражает скорее благие

пожелания, чем действительность: иерархия каталогов одинакова только для дистрибутивов, связанных единством происхождения, а исторически сложившиеся различия создают опасность несовместимости разных дистрибутивов. Краткое описание основных каталогов сведено в таблицу 1.

Таблица 1 – Краткое описание основных каталогов

Каталог	Назначение
/bin	Основные программы, необходимые для работы в системе: командные оболочки, файловые утилиты и т. п.
/sbin	Команды для системного администрирования, а также программы, выполняемые в процессе загрузки
/boot	Файлы, необходимые для загрузки системы (образ ядра)
/home	Домашние каталоги пользователей, кроме root
/dev	Файлы устройств
/etc	Файлы настроек: стартовые сценарии, конфигурационные файлы графической системы и различных приложений
/lib	Системные библиотеки, необходимые для основных программ, и модули ядра
/lost+found	Восстановленные после аварийного размонтирования части файловой системы
/media	Сюда обычно монтируются съемные носители: компакт-диски, flash-накопители
/mnt	Временные точки монтирования жестких дисков. Использовать этот каталог необязательно: подмонтировать файловую систему можно к любому другому каталогу
/opt	Дополнительные пакеты программ. Если программа, установленная сюда, больше не нужна, то достаточно удалить ее каталог без обычной процедуры деинсталляции
/proc	Виртуальная файловая система, дающая доступ к информации ядра (например, выведите на экран файл/proc/cpuinfo). Другие файлы в этом каталоге в каждый момент времени содержат информацию о выполняющихся в этот момент программах
/root	Домашний каталог суперпользователя. Домашние каталоги всех остальных могут находиться в отдельном разделе, но /root должен быть в корневой файловой системе, чтобы администратор всегда мог войти в систему для ремонтных работ
/tmp	Временные файлы
/var	Часто меняющиеся данные: системные журналы и протоколы приложений, замки, почтовые ящики, очереди печати и т. п.
/usr	Практически все остальное: программы, исходные коды, документация. Сюда по умолчанию устанавливаются новые программы

С точки зрения UNIX-подобных ОС, файл представляет собой поток или последовательность байтов. Такой подход позволяет распространить понятие файла на множество ресурсов не только локального компьютера, но и удаленного, связанного с локальной сетью любого рода. Доступ к любому такому ресурсу осуществляется через универсальный интерфейс, благодаря чему запись данных в файл, отправка их на физическое устройство или обмен

ими с другой работающей программой происходят аналогично. Это очень упрощает организацию данных и обмен ими.

В ОС Linux можно выделить следующие типы файлов:

– обычные файлы – последовательность байтов (текстовые документы, исполняемые программы, библиотеки и т. п.);

– каталоги – именованные наборы ссылок на другие файлы;

– файлы физических устройств, подразделяющиеся на:

а) файлы блочных устройств, драйверы которых буферизуют ввод-вывод с помощью ядра и файлы байт-ориентированных, или символьных устройств, позволяющих связанным с ними драйверам выполнять буферизацию собственными средствами;

б) символические ссылки (symlink, symbolic link);

в) именованные каналы (named pipes);

г) гнезда (sockets).

Задание

1 Для выполнения данной работы будем использовать ранее установленный Linux Ubuntu. Запускаем Linux. После прохождения идентификации включаем терминал.

2 Для работы в терминале Ubuntu требуются права пользователя root, но, к сожалению, по умолчанию, он недоступен, поэтому для выполнения некоторых (не всех) команд надо писать sudo <команда>, и подтверждать свои права вводом пароля. И не пугайтесь того, что его не видно в терминале.

3 Для получении справки о дополнительных возможностях некоторых программ следует набрать <команда> --help.

4 Выполните следующие команды:

– date;

– hwclock;

– uname;

– history;

– clear;

– ls.

5 Создайте нового пользователя, при помощи терминала Ubuntu, и введите его в группу admin. Создайте пароль пользователю. Войдите под ним в систему. Процесс создания и ввода в группу внесите в отчет.

Контрольные вопросы

1 Что такое терминал?

2 Перечислите основные системные каталоги.

3 Расскажите о типах файлов в ОС Linux.

3 Лабораторная работа № 3. Изучение файловой системы и функций по обработке и управлению данными

Цель работы: изучение структуры файловой системы ОС Linux, изучение команд создания, удаления, модификации файлов и каталогов, функций манипулирования данными.

Файловая структура системы Linux. В операционной системе Linux файлами считаются обычные файлы, каталоги, а также специальные файлы, соответствующие периферийным устройствам (каждое устройство представляется в виде файла). Доступ ко всем файлам однотипный, в том числе, и к файлам периферийных устройств. Такой подход обеспечивает независимость программы пользователя от особенностей ввода/вывода на конкретное внешнее устройство.

Файловая структура Linux имеет иерархическую древовидную структуру. В корневом каталоге размещаются другие каталоги и файлы, включая пять основных каталогов:

1) bin – большинство выполняемых командных программ и shell-процедур;
2) tmp – временные файлы;
3) usr – каталоги пользователей (условное обозначение);
4) etc – преимущественно административные утилиты и файлы;
5) dev – специальные файлы, представляющие периферийные устройства; при добавлении периферийного устройства в каталог /dev должен быть добавлен соответствующий файл (черта / означает принадлежность корневому каталогу).

Текущий каталог – это каталог, в котором в данный момент находится пользователь. При наличии прав доступа, пользователь может перейти после входа в систему в другой каталог. Текущий каталог обозначается точкой (.); родительский каталог, которому принадлежит текущий, обозначается двумя точками (..).

Полное имя файла может включать имена каталогов, включая корневой, разделенных косой чертой, например: /home/student/file.txt. Первая косая черта обозначает корневой каталог, и поиск файла будет начинаться с него, а затем в каталоге home, затем в каталоге student.

Один файл можно сделать принадлежащим нескольким каталогам. Для этого используется команда ln (link):

ln <имя файла 1> <имя файла 2>.

Имя первого файла – это полное составное имя файла, с которым устанавливается связь; имя второго файла – это полное имя файла в новом каталоге, где будет использоваться эта связь. Новое имя может не отличаться от старого. Каждый файл может иметь несколько связей, т. е. он может использоваться в разных каталогах под разными именами. Команда ln с аргументом -s создает символическую связь:

ln -s <имя файла 1> <имя файла 2>.

Здесь имя второго файла является именем символической связи. Символическая связь является особым видом файла, в котором хранится имя файла, на который символическая связь ссылается. Linux работает с символической связью не так, как с обычным файлом – например, при выводе на экран содержимого символической связи появятся данные файла, на который эта символическая связь ссылается.

В Linux различаются 3 уровня доступа к файлам и каталогам:

1) доступ владельца файла;

2) доступ группы пользователей, к которой принадлежит владелец файла;

3) остальные пользователи.

Для каждого уровня существуют свои байты атрибутов, значение которых расшифровывается следующим образом:

r – разрешение на чтение;

w – разрешение на запись;

x – разрешение на выполнение;

– – отсутствие разрешения.

В домашнем каталоге пользователь имеет полный доступ к файлам (READ, WRITE, EXECUTE; r, w, x).

Атрибуты файла и доступ к нему можно изменить командой

chmod <коды защиты> <имя файла>.

Коды защиты могут быть заданы в числовом или символьном виде. Для символьного кода используются:

знак плюс (+) – добавить права доступа;

знак минус (–) – отменить права доступа;

r, w, x – доступ на чтение, запись, выполнение.

Коды защиты в числовом виде могут быть заданы в восьмеричной форме. Для контроля установленного доступа к своему файлу после каждого изменения кода защиты нужно проверять свои действия с помощью команды ls -l.

Задание

1 Ознакомтесь с файловой структурой ОС. Изучите команды работы с файлами.

2 Используя команды ОС, создайте два текстовых файла.

3 Полученные файлы объедините в один файл и его содержимое выведите на экран.

4 Создайте новую директорию и переместите в нее полученные файлы.

5 Выведите полную информацию обо всех файлах и проанализируйте уровни доступа.

6 Добавьте для всех трех файлов право выполнения членам группы и остальным пользователям.

7 Просмотрите атрибуты файлов.

8 Создайте еще один каталог.

9 Установите дополнительную связь объединенного файла с новым каталогом, но под другим именем.

10 Создайте символическую связь.

11 Сделайте текущим новый каталог и выведите на экран расширенный список информации о его файлах.

12 Произведите поиск заданной последовательности символов в файлах текущей директории и получите перечень соответствующих файлов.

13 Получите информацию об активных процессах и имена других пользователей.

Контрольные вопросы

1 Что считается файлами в ОС Linux?

2 Объясните назначение связей с файлами и способы их создания.

3 Что определяет атрибуты файлов и каким образом их можно просмотреть и изменить?

4 Какие методы создания и удаления файлов, каталогов вы знаете?

5 В чем заключается поиск по шаблону?

6 Какой командой можно получить список работающих пользователей и сохранить его в файле?

4 Лабораторная работа № 4. Процессы в операционной системе Linux

Цель работы: изучение процессов в операционной системе Linux.

Лабораторная работа посвящена процессам операционной системы Linux. Поскольку администрирование операционной системы в конечном счете сводится к управлению процессами. Каждый раз при запуске программы на выполнение создается процесс. Процессом называется выполняемая в данный момент программа или ее потомки. Каждый процесс запускается от имени какого-то пользователя. Процессы, которые стартовали при загрузке, обычно выполняются от имени пользователей root или nobody.

Каждый пользователь может управлять поведением процессов, им запущенных. При этом пользователь root может управлять всеми процессами – как запущенными от его имени, так и процессами, порожденными другими пользователями операционной системы. Управление процессами осуществляется с помощью утилит, а также посредством некоторых команд командной оболочки shell.

Каждый процесс в системе имеет уникальный номер – идентификационный номер процесса (Process Identification, PID). Этот номер используется ядром операционной системы, а также некоторыми утилитами для управления процессами.

Выполнение процесса на переднем плане и в фоновом режиме.

Процессы могут выполняться на переднем плане (foreground) — режим по умолчанию и в фоновом режиме (background). На переднем плане в каждый момент для текущего терминала может выполняться только один процесс. Однако пользователь может перейти в другой виртуальный терминал и запустить на выполнение еще один процесс, а на другом терминале еще один и т. д. Процесс переднего плана – это процесс, с которым взаимодействует пользователь. Этот процесс получает информацию с клавиатуры (стандартный ввод) и посылает результаты на экран (стандартный вывод).

Фоновый процесс после своего запуска благодаря использованию специальной команды командной оболочки отключается от клавиатуры и экрана, т. е. не ожидает ввода данных со стандартного ввода и не выводит информацию на стандартный вывод, а командная оболочка не ожидает окончания запущенного процесса, что позволяет пользователю немедленно запустить еще один процесс.

Обычно фоновые процессы требуют очень большого времени для своего завершения и не требуют вмешательства пользователя во время существования процесса. К примеру, компиляция программ или архивирование большого объема информации – кандидаты номер один для перевода процесса в фоновый режим.

Процессы могут быть отложенными. Отложенный процесс – это процесс, который в данный момент не выполняется и временно остановлен. После того как процесс остановлен, он может быть продолжен как на переднем плане, так и в фоновом режиме. Возобновление приостановленного процесса не изменит его состояния – при возобновлении он начнется с того места, на котором был приостановлен.

Для выполнения программы в режиме переднего плана достаточно просто набрать имя программы в командной строке и запустить ее на выполнение. После этого вы можете работать с программой.

Для запуска программы в качестве фонового процесса достаточно набрать в командной строке имя программы и в конце добавить знак амперсанта (&), отделенный пробелом от имени программы и ее параметров командной строки, если таковые имеются. Затем программа запускается на выполнение.

Задание

1 Запустите программу shell в фоновом режиме с подавлением потока вывода.

2 Запустите программу shell на переднем плане с подавлением потока вывода. Приостановите выполнение программы. Заново запустите программу shell с теми же параметрами и завершите ее выполнение.

3 Запустите программу shell на переднем плане без подавления потока вывода. Приостановите выполнение программы. Заново запустите программу shell с теми же параметрами, и завершите ее выполнение.

4 Проверьте состояния процессов, воспользовавшись командой jobs.

5 Переведите процесс, который у вас выполняется в фоновом режиме на передний план и остановите его.

6 Переведите любой ваш процесс с подавлением потока вывода в фоновый режим.

7 Проверьте состояния процессов, воспользовавшись командой jobs. Обратите внимание, что процесс стал выполняющемся (Running) в фоновом режиме.

8 Запустите процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала.

9 Закройте окно и заново запустите консоль. Убедитесь, что процесс продолжил свою работу.

Контрольные вопросы

1 Объясните, что произойдет, если запустить программу в фоновом режиме без подавления потока вывода.

2 Объясните разницу между действием сочетаний клавиш ^Z и ^C.

3 Опишите, что значит каждое поле вывода команды jobs.

4 Назовите главное отличие утилиты top от ps.

5 Лабораторная работа № 5. Организация ввода-вывода в ОС Linux

Цель работы: изучение средств управления потоками ввода/вывода в ОС Linux.

Работа со стандартными потоками ввода, вывода и ошибок. Любая программа – это «автомат», предназначенный для обработки данных: получая на входе одну информацию, она в результате работы выдает другую. Хотя входящая и/или выходящая информация может быть и нулевой, т. е. попросту отсутствовать. Те данные, которые передаются программе для обработки – это ее ввод, то, что она выдает в результате работы, – вывод. Организация ввода и вывода для каждой программы – это задача операционной системы.

Каждая программа работает с данными определенного типа: текстовыми, графическими, звуковыми и т. п. Как, наверное, уже стало понятно, основной интерфейс управления системой в Linux – это терминал, который предназначен для передачи текстовой информации от пользователя системе и обратно. Поскольку ввести с терминала и вывести на терминал можно только текстовую информацию, то ввод и вывод программ, связанных с терминалом, тоже должен быть текстовым. Однако необходимость оперировать с текстовыми данными не ограничивает возможности управления системой, а, наоборот, расширяет их. Человек может прочитать вывод любой программы и разобраться, что происходит в системе, а разные программы оказываются совместимыми между собой, поскольку используют один и тот же вид представления данных – текстовый.

Команды и сценарии могут получать входные данные двумя способами: из стандартного входного потока (связан с клавиатурой) или из файла. Аналогичное разделение существует и при выводе данных: результаты работы команды или сценария по умолчанию направляются на экран терминала, но можно перенаправить их в файл. Если в процессе работы возникают ошибки,

сообщения о них тоже отображаются на экране, поток ошибок также можно перенаправить в файл.

Рассмотрим сначала пару команд, с помощью которых можно организовать ввод/вывод.

Команды вывода на стандартное устройство вывода. Linux предоставляет несколько команд для вывода сообщений в стандартный поток вывода:

echo – вывести строку в стандартный поток вывода;

printf – вывести форматированный текст в стандартный поток вывода;

yes – выводить повторяющийся текст в стандартный поток вывода;

seq – вывести последовательность чисел в стандартный поток вывода;

clear – очистить экран или окно.

Например, при использовании команды echo, если указать управляющий символ \c, то по завершении вывода не будет осуществлен переход в новую строку:

```
$ echo "Как вас зовут?\c"
```

В строке также можно вычислять значения переменных интерпретатора shell и даже других команд.

Большое число утилит используют только стандартные потоки. Для таких программ оболочка позволяет независимо перенаправлять потоки ввода/вывода. Например, можно подавить вывод сообщений об ошибках, установить ввод или вывод из файла.

То есть при вызове команд можно указывать, откуда следует принимать входные данные и куда необходимо направлять выходные данные, а также сообщения об ошибках. По умолчанию, если не указано иное, подразумевается работа с терминалом: данные вводятся с клавиатуры и выводятся на экран. Но интерпретатор shell располагает механизмом переадресации, позволяющим ассоциировать стандартные потоки с различными файлами. При этом во время перенаправления стандартного потока ошибок следует указывать дескриптор файла. Для потоков ввода и вывода делать это не обязательно.

Однако, в отличие от DOS при создании программного канала между двумя процессами ОС UNIX/Linux запускает оба процесса одновременно и осуществляет передачу информации через системный буфер (без промежуточной записи на жесткий диск). Таким образом, программные каналы в ОС UNIX/Linux являются весьма эффективным способом обмена. В случае переполнения системного буфера (например, если «передающая» программа выдает информацию в канал быстрее, чем ее может обработать «принимающая» программа) ОС автоматически приостанавливает тот процесс, который осуществляет запись в канал до освобождения буфера.

Задание

Разработать скрипт вычисления суммы двух чисел. Исходные данные запрашивать у пользователя, запустившего скрипт на выполнение. Результаты работы скрипта вывести на консоль.

Примечание – Скрипт – исполняемый файл, содержащий последовательность команд и операторов. Запуск скрипта с именем name_script в консоли выполняется одним из двух способов: с помощью команды sh - $ sh name_script; путем назначения файлу права выполнения и запуска, например, из текущей директории - $ ./name_script.

Контрольные вопросы

1 Что такое скрипт?
2 Опишите способы запуска скриптов.
3 Объясните следующую команду sh - $ sh name_script.
4 Что такое назначение файлу права выполнения и запуска?

6 Лабораторная работа № 6. Создание и выполнение командных файлов в пользовательской среде ОС Linux

Цель работы: изучение методов создания и выполнения командных файлов на языке Shell-интерпретатора.

Shell является языком программирования, который применяется для написания командных файлов (shell-файлов). Командные файлы также называются скриптами и сценариями. Shell-файл содержит одну или несколько выполняемых команд (процедур), а имя файла в этом случае используется как имя команды.

Для обозначения переменных Shell используется последовательность букв, цифр и символов подчеркивания; переменные не могут начинаться с цифры. Присваивание значений переменным проводится с использованием знака = , например, PS2 = '<' . Для обращения к значению переменной перед ее именем ставится знак $.

Переменные -; ?; #; $; ! устанавливаются только Shell. Они позволяют с помощью команды echo получить следующую информацию:

– – текущие флаги интерпретатора (установка флагов может быть изменена командой set);

– число аргументов, которое было сохранено интерпретатором при выполнении какой-либо команды;

? – код возврата последней выполняемой команды;

$ – числовой идентификатор текущего процесса PID;

! – PID последнего фонового процесса.

Задание

Составьте и выполните shell-программы, включающие следующие действия:

1) вывод на экран списка параметров командной строки с указанием номера каждого параметра;

2) присвоение переменным A, B и C значений 10, 100 и 200, вычисление и вывод результатов по формуле D = (A*2 + B/3)*C;

3) формирование файла со списком файлов в домашнем каталоге, вывод на

экран этого списка в алфавитном порядке и общего количества файлов;

4) переход в другой каталог, формирование файла с листингом каталога и возвращение в исходный каталог;

5) запрос и ввод имени пользователя, сравнение с текущим логическим именем пользователя и вывод сообщения: верно/неверно;

6) запрос и ввод имени файла в текущем каталоге и вывод сообщения о типе файла;

7) циклическое чтение системного времени и очистка экрана в заданный момент;

8) циклический просмотр списка файлов и выдача сообщения при появлении заданного имени в списке.

Контрольные вопросы

1 Какое назначение имеют shell-файлы?
2 Как создать shell-файл и сделать его выполняемым?
3 Какие типы переменных используются в shell-файлах?
4 В чем заключается анализ цепочки символов?
5 Какие встроенные команды используются в shell-файлах?
6 Как производится управление программами?
7 Назовите операторы создания циклов.

7 Лабораторная работа № 7. Удаленный доступ в Linux

Цель работы: получить начальные навыки работы с удаленным хостом по протоколу ssh.

Средства удаленного управления серверами широко используются как в локальных, так и в глобальных сетях. Основное назначение таких средств – организация канала передачи (в общем случае – через шлюз) команд управления сервером и возврат клиенту результата выполнения этих команд. Поэтому одними из основных требований к программам удаленного управления являются прозрачность для пользователя и небольшой трафик. Это позволяет централизованно управлять территориально распределенными узлами с одного рабочего места или представлять доступ удаленным терминальным клиентам по медленным линиям связи.

Сетевые шлюзы. Подключение локальных сетей к Интернет обычно реализуется через коммуникационный сервер – шлюз (gateway). Такой сервер обычно работает под управлением какой-нибудь версии UNIX (FreeBSD, Debian GNU/Linux, RHEL и т. п.) и имеет как минимум два сетевых интерфейса: один внутренний, к которому подключена ЛВС, второй – внешний, обращенный в Интернет. На шлюзе настроена маршрутизация, подняты сервисы NAT и прокси.

Для управления используются как символьные протоколы (telnet, rlogin, ssh), так и бинарные, поддерживающие графические возможности. Для выде-

ленных серверов графические средства как правило не используются, поскольку такие серверы не подразумевают использование их как рабочих станций. Это означает, что нет необходимости выделять ресурсы для графического пользовательского интерфейса.

Текстовые протоколы tenet и rlogin просты и функциональны, но небезопасны. Исходя из указанных соображений в текущей работе предполагается освоение удаленного управления UNIX-сервером по протоколу ssh. Следует отметить, что протокол ssh поддерживает и работу с графическим режимом (тунеллирование X-сервера). Более того, протокол ssh позволяет туннеллировать любой сетевой трафик, использующий в качестве транспорта протокол TCP.

Для управления сервером по протоколу ssh необходима его поддержка сервером и клиентское ssh-приложение. UNIX-серверы стандартно поддерживают протокол ssh. В качестве клиента, как правило, используется OpenSSH (вызывается командой ssh). Для Windows имеются клиенты разных производителей, наиболее популярные PuTTY и SecureCRT. В лабораторной работе предполагается использование клиента OpenSSH.

Подключение к серверу. Для подключения к серверу необходимо выполнить команду ssh, указав в качестве параметров имя пользователя и имя или сетевой адрес сервера:

aag@stilo:~> ssh -l student edu.asoiu // -l student - логин; edu.asoiu - ssh-сервер

Или так:

aag@stilo:~> ssh student@edu.asoiu

Если вы выполните ssh, не задав имя пользователя, то серверу будет отправлено имя текущего локального пользователя.

При первом ssh подключении к удалённой машине, вы увидите подобное сообщение:

The authenticity of host 'edu.asoiu' can't be established.
DSA key fingerprint is 94:68:3a:3a:bc:f3:9a:9b:01:5d:b3:07:38:e2:11:0c.
Are you sure you want to continue connecting (yes/no)?

Введите yes для продолжения. При этом сервер будет добавлен в ваш список известных серверов, о чём говорит следующее сообщение:

Warning: Permanently added 'edu.asoiu' (DSA) to the list of known hosts.

Затем будет выведено приглашение для ввода пароля удалённого компьютера. После авторизации на сервере пользователь попадает в оболочку UNIX (как правило это одна из версий shell, зависит от настроек сервера) и может приступать к вводу команд.

Задание

С помощью команды ssh определите тип операционной системы, аппаратную платформу и версию ядра сервера (команда uname с соответствующими

параметрами) без загрузки оболочки сервера.

С помощью ssh-клиента подключитесь к серверу edu.asoi с учетной записью student.

Определите, какой каталог является текущим на удаленном сервере. Если он отличается от /home/student, то выполните переход в /home/student.

Определите, какие пользователи, в каких терминалах и с каких IP-адресов подключены к серверу (команда who).

В текущем каталоге создайте каталог (например, ivanov).

Запустите файловый менеджер mc, просмотрите содержимое текущего каталога.

Завершите работу с файловым менеджером mc.

Завершите сеанс ssh.

Не входя в сеанс ssh, загрузите в ранее созданный вами на сервере каталог произвольный файл (команда scp).

Не входя в сеанс ssh, просмотрите содержимое серверного каталога /home/student .

Контрольные вопросы

1 Назначение команды ssh.
2 Опишите способ подключения к серверу с помощью ssh-клиента.
3 Как определить, какой каталог является текущим на удаленном сервере?
4 Назначение команды who.

8 Лабораторная работа № 8. Управление пользователями и обеспечение безопасности в ОС Linux

Цель работы: приобрести опыт запуска и настройки общесистемных сервисов (конфигурирования системы), управления пользователями и обеспечения безопасности.

Для создания группы используется команда addgroup, например, addgroup g301.

Для создания пользования используется команда adduser, например, adduser anton passwd anton. При создании группы, пользователя в файлы passwd, group каталога /etc вносится информация о пользователе и группе. Одна учетная запись соответствует одному пользователю и одной группе. Учетная запись пользователя содержит следующую информацию:

– имя пользователя;
– пароль в закодированном виде;
– целочисленный идентификатор пользователя;
– целочисленный идентификатор группы;
– комментарий;

– каталог пользователя;

– интерпретатор команд. Например, anton:*:100:1000::/home/usr:/bin/bash.

Учетная запись группы содержит следующую информацию: имя группы; пароль в закодированном виде; целочисленный идентификатор группы; список имен пользователей группы. Например, g301:*:1000: anton.

Чтобы удалить пользователя или группу используются команды: userdel, delgroup соответственно.

Чтобы изменить пользовательскую, групповую принадлежность, используются команды chown, chgrp. Например, chown user file1 chgrp it402 file1.

Изменение пароля осуществляется командой passwd, например, passwd anton. Введите пароль, повторите пароль, тем самым вы измените пароль указанному пользователю.

Права доступа к файлам и каталогам. Права доступа к файлам разделяются на три категории: права владельца файла, права группы, связанной с файлом, и права всех остальных пользователей. Каждая категория имеет свой набор прав доступа к файлу, которые обеспечивают возможность чтения из файла, записи в файл и его выполнения (или, наоборот, запрещают эти действия). Права доступа называются также режимом доступа к файлу. Режимы доступа к файлу устанавливаются с помощью команды chmod.

Задание

1 Добавьте в систему нового пользователя.

2 Войдите в систему под пользователем root.

3 Добавьте в систему пользователя с именем student (используйте команды useradd и passwd):

Useradd student Passwd student

4 После ввода команды passwd student, на экране появится сообщение: Enter new password:

Вы должны задать пароль для пользователя (не менее 5 символов) и нажать <Enter>. Обратите внимание, что при вводе пароля, он не отображается на экране. После этого выведется сообщение:

Re-enter new password:

Вы должны ещё раз ввести тот же самый пароль, который был вами задан для пользователя student. После нажатия на клавишу <Enter> на экран выведется: Password changed.

Итак, вы зарегистрировали нового пользователя в системе и задали ему пароль, теперь можно зайти под этим пользователем, воспользовавшись командой login: [home/dsl]# login Box login: student Password:

После ввода пароля вы увидите приглашение: student@tty0[/]$

Введите команду whoami и убедитесь в том, что вы зашли в систему под пользователем student. Для завершения работы с пользователем воспользуйтесь командой logout.

Контрольные вопросы

1 Какие основные каталоги содержатся в корневом каталоге в Linux?

2 Какую команду необходимо использовать, чтобы просмотреть содержимое каталога?

3 Как обозначаются родительский каталог и домашний каталог пользователя? Какая команда используется для навигации по файловой системе?

4 Как запустить текстовый редактор vi? Какие клавиши нужно нажать, чтобы начать вводить текст в этом текстовом редакторе? Как сохранить текст и выйти из программы vi?

5 Как удалить всю строку целиком в текстовом редакторе vi? Какие ещё команды vi для работы с текстом вы знаете?

6 Как просмотреть содержимое текстового файла?

7 Какой командой осуществляется поиск в файле и вывод на экран строк, содержащих заданный текст?

8 Какие существуют права доступа к файлам и каталогам? Как задать права для файла, чтобы он был доступен только для чтения для всех пользователей?

9 Как войти в систему Linux? Как добавить, удалить нового пользователя?

10 Как завершить работу с системой Linux?

11 Для чего предназначена программа Midnight Commander?

12 Почему нужно быть особенно осторожным при работе в системе Linux под пользователем root?

13 Что означают права доступа к файлу, обозначенные числом 762? Какие команды нужно знать, чтобы добавить пользователя в систему? Как удалить пользователя в Linux?

9 Лабораторная работа № 9. Администрирование DNS-сервера в ОС Linux

Цель работы: изучить понятие доменных имен, разобраться с назначением и принципом функционирования службы доменных имен (DNS), ознакомиться с базовыми понятиями протокола DNS, научиться устанавливать и настраивать простейший вариант DNS сервера.

Домен – область (ветвь) иерархического пространства доменных имён сети Интернет, которая обозначается уникальным доменным именем.

Доменное имя – символьное имя домена. Должно быть уникальным в рамках одного домена. Полное имя домена состоит из имён всех доменов, в которые он входит, разделённых точками. Например, полное имя www.tu-bryansk.ru. (с точкой в конце) обозначает домен третьего уровня www, который входит в домен второго уровня tu-bryansk, который входит в домен .ru, который входит в корневой домен. Доменное имя служит для адресации узлов

сети Интернет и расположенных на них сетевых ресурсов (веб-сайтов, серверов электронной почты, сетевых сервисов) в удобной для человека форме.

Доменная зона – совокупность доменных имён определённого уровня, входящих в конкретный домен. Например, зона ctam.tu-bryansk.ru. означает все доменные имена третьего уровня в этом домене. Термин «доменная зона» в основном применяется в технической сфере, при настройке DNS-серверов (поддержание зоны, делегирование зоны, трансфер зоны).

Для обеспечения уникальности и защиты прав владельцев доменные имена 1-го и 2-го (в отдельных случаях и 3-го) уровней можно использовать только после их регистрации, которая производится уполномоченными на то регистраторами. Сведения о владельце (администраторе) того или иного регистрируемого домена общедоступны. Их можно узнать, воспользовавшись службой «whois» – например, http://www.ripn.net:8080/nic/whois.

Домены верхнего уровня общего назначения:

.aero – для субъектов авиатранспортной индустрии;

.biz – только коммерческие организации;

.cat – для использования каталанским языковым и культурным сообществом;

.com – коммерческие организации (без ограничений);

.coop – кооперативы;

.edu – высшие учебные заведения, признаваемые в качестве таковых Департаментом образования США;

.info – информационные ресурсы (без ограничений);

.jobs – кадровые агентства;

.mobi – для продавцов и поставщиков мобильного контента и услуг, связанных с мобильной связью;

.museum – музеи;

.name – физические лица;

.net – организации, имеющие отношение к функционированию Интернета (без ограничений);

.org – некоммерческие организации (без ограничений);

.pro – сертифицированные профессионалы и смежные темы;

.travel – для субъектов туристического бизнеса.

Домены верхнего уровня, назначаемые странам.

Для удобства распределения и назначения доменных имен для каждой из стран были выделены собственные (в основном двухсимвольные) домены верхнего уровня. Правда, это вовсе не означает обязательную привязку серверов в данных доменах к их географическому расположению. Примеры доменов первого уровня для стран:

.au – Australia (Австралия);

.be – Belgium (Бельгия);

.ru – Russia (Россия);

.ua – Ukraine (Украина);

.uk – United Kingdom (Англия).

Служба трансляции имен в Интернете.

Первоначально преобразование между доменными и IP-адресами производилось с использованием специального текстового файла DHOSTS.TXT, который составлялся централизованно и обновлялся на каждой из машин сети вручную. С ростом Сети возникла необходимость в эффективном, автоматизированном механизме, которым и стала DNS (Domain Name System) – система доменных имен.

Примечание – На самом деле на каждой сетевой машине имеется текстовый файл hosts (Windows – %windir%\system32\drivers\etc\hosts; *nix – /etc/hosts), в котором можно самостоятельно сопоставлять с некоторым IP-адресом доменные имена. Правда, эти действия валидны только для текущей машины.

Функции DNS. Существуют два принципиально разных способа идентификации хостов: с помощью имен и с помощью IP-адресов. Имя хоста удобно для людей в силу своей мнемоничности, а IP-адрес, являющийся компактной числовой величиной фиксированного размера, проще обрабатывать прикладными программами и маршрутизаторами. Для того чтобы установить связь между этими двумя идентификаторами, используется система доменных имен. DNS представляет собой, с одной стороны, базу данных, распределенную между иерархически структурированными серверами имен, и, с другой стороны, протокол прикладного уровня, организующий взаимодействие между хостами и серверами имен для выполнения операций преобразования.

DNS функционирует на принципе делегирования полномочий. Каждая машина либо знает ответ на вопрос, либо знает кого спросить. При правильном функционировании система замкнута, т. е. если запрошенная информация имеется у кого-либо, то она будет найдена и сообщена клиенту, либо, если вопрос не имеет ответа, клиент получит сообщение о невозможности получения ответа на вопрос.

Обратный DNS-запрос (Reverce DNS). DNS используется в первую очередь для преобразования символьных имён в IP-адреса, но он также может выполнять обратный процесс. Для этого используются уже имеющиеся средства DNS. Дело в том, что с записью DNS могут быть сопоставлены различные данные, в том числе и какое-либо символьное имя. Существует специальный домен in-addr.arpa., записи в котором используются для преобразования IP-адресов в символьные имена. Например, для получения DNS-имени для адреса 192.168.128.5 можно запросить у DNS-сервера запись 5.128.168.192.in-addr.arpa, и тот вернёт соответствующее символьное имя. Обратный порядок записи частей IP-адреса объясняется тем, что в IP-адресах старшие биты расположены в начале, а в символьных DNS-именах старшие (находящиеся ближе к корню) части расположены в конце.

Одна из проблем состоит в том, что обратную зону можно выделить только на сетях класса A, B или C (на 16777216, 65536 или 256 адресов соответственно) и никак иначе (маски здесь не работают).

Установка и настройка DNS сервера Ubuntu. Для начала нужно установить самые последние обновления системы:

```
sudo apt-get update
sudo apt-get upgrade.
```

После этого установим службу DNS сервера Bind9 при помощи команды sudo aptitude install bind9.

Задание

1 Выясните примерное географическое месторасположение корневых серверов имен (можно воспользоваться одним из сервисов whois).

2 Настройте DNS сервер на базе Windows Server 2016 (в качестве forward сервера рекомендуется использовать DNS-сервера 192.168.128.1 или 192.168.128.5). Настройка и управление DNS-сервером осуществляется через консоль управления dnsmgmt.

3 Проверьте работоспособность настроенного сервиса.

4 При помощи сниффера wireshark исследуйте механизм работы утилиты nslookup.

5 Сделайте выводы. Подготовьте отчет с результатами проделанной работы.

Контрольные вопросы

1 Что такое DHCP сервер?
2 Что такое DNS сервер?
3 Способы настройки динамического обновления зон DNS сервером.
4 Какие вы знаете зоны DNS сервера.

10 Лабораторная работа № 10. Маршрутизация в ОС Linux. Межсетевое экранирование в Linux

Цель работы: получить сведения о маршрутизации и научиться добавлять маршруты в таблицу маршрутизации.

В сетях, основанных на протоколе IP, концепция маршрутизации является одной из важных. Она создает или разбивает сеть. Неправильная конфигурация маршрутизации способна вывести из строя сеть.

Маршрутизация – технология определения пути доставки (маршрута) пакетов.

Каждая операционная система, поддерживающая стек TCP/IP, имеет маршрутизатор и таблицу маршрутизации.

Таблица маршрутизации используется только тогда, когда определяется, как доставлять пакеты.

Маршрутизация должна быть сконфигурирована корректно на обоих концах связи и на каждом участке между ними.

Для определения пути доставки пакета используется таблица маршрутизации. Пример таблицы маршрутизации можно получить командой route с параметром print (рисунок 1).

В общем случае для маршрутизации используется следующий алгоритм. Из пакета извлекается IP-адрес назначения пакета и производится попытка

сопоставить его с адресом назначения (Сетевой адрес) каждого элемента таблицы маршрутизации, пока не найдется наилучшее совпадение. Если совпадений не найдено, то пакет удаляется и отправителю пакета может отправиться сообщение об ошибке. Сравнение производится с тремя порциями информации: Сетевой адрес (Network Destination), Маска сети (Netmask) и IP-адрес назначения пакета.

Активные маршруты:

Сетевой адрес	Маска сети	Адрес шлюза	Интерфейс	Метрика
Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.4.1	192.168.4.7	1
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.4.0	255.255.255.0	192.168.4.7	192.168.4.7	1
192.168.4.7	255.255.255.255	127.0.0.1	127.0.0.1	1
192.168.4.255	255.255.255.255	192.168.4.7	192.168.4.7	1
224.0.0.0	224.0.0.0	192.168.4.7	192.168.4.7	1
255.255.255.255	255.255.255.255	192.168.4.7	192.168.4.7	1
Основной шлюз:	192.168.1.1			

Рисунок 1 – Пример таблицы маршрутизации

В основном, производится побитная операция AND между IP-адресом получателя и Маской сети (Netmask): если полученное значение равно Сетевому адресу (Network Destination), то считается, что совпадение найдено.

Обычно VRF (Virtual Routing and Forwarding, VPN Routing and Forwarding) используется совместно с MPLS, в терминологии Cisco называется VRF-Lite. Суть этой технологии в том, что маршрутная информация, принадлежащая различным классам (например, маршруты одного клиента), изолируется друг от друга.

Типы маршрутов, таблицы маршрутизации и PBR в Linux. Прежде чем понять суть происходящего, познакомимся с некоторыми отличительными чертами сетевого стека Linux.

Первый отличительный момент – это специальные типы маршрутов. Когда IP-пакет приходит с какого-нибудь интерфейса, надо определить, адресован ли он этому хосту, или другому. Определяется это довольно элегантно – просто для адреса назначения ищется нужный маршрут в таблицах маршрутизации. Если пакет попадает на маршрут типа «local», значит он адресован непосредственно хосту, если нет, то значит, его надо маршрутизировать дальше (при этом дальнейший маршрут уже известен) или сделать что-то ещё, в зависимости от типа маршрутов. На данный момент поддерживается несколько типов маршрутов (подробнее о них можно посмотреть в справке ip-route). Нас в данный момент интересуют только маршруты следующих типов.

unicast – обычный маршрут.

local – адрес назначения находится на данном хосте. После того, как выяснится, что пакет попадает на этот маршрут, будет производиться поиск подходящего сокета для него.

broadcast – широковещательный маршрут. Для входящих пакетов, попадающих на этот маршрут, практически нет отличий от маршрута local, за исключением дополнительных проверок на игнорирование широковещательных пакетов. Для исходящих же есть небольшое отличие: в заголовке канального уровня также выставляется широковещательный адрес назначения при использовании широковещательных сетей.

unreachable – запрещающий маршрут. Для пакетов, попадающих на этот маршрут будет отослан отправителю icmp-пакет с сообщением о недоступности.

prohibit – подобен типу unreachable, только сообщение будет отправлено другое.

blackhole – пакет на этом маршруте будет молча отброшен.

Таким образом, для маршрутизации транзитных пакетов нам достаточно наличия маршрута типа unicast, а для того, чтобы хост мог отвечать на пакеты, нужны ещё маршруты типов local и, опционально, broadcast. Ещё следует учесть то, что нам также нужны маршруты direct-connected сетей для того, чтобы обеспечить связность с соседними маршрутизаторами.

Маршруты сгруппированы в таблицы маршрутизации. По-умолчанию изначально в системе присутствуют три таблицы:

1) local (255) – в этой таблице находятся локальные и широковещательные маршруты. Эта таблица обслуживается автоматически и генерируется на основе адресов, назначенных интерфейсам;

2) main (254) – основная таблица маршрутизации. Автоматически в неё добавляются direct-connected маршруты. Если в параметрах утилиты IP не указана таблица маршрутизации, то подразумевается таблица main;

3) default (253) – таблица для маршрутов по-умолчанию.

Имена таблиц хранятся в файле /etc/iproute2/rt_tables. Под номер таблицы отдано 32 бита, но максимальное количество таблиц в данный момент жёстко ограничено числом 256.

Таблица, в которой надо искать маршруты, определяется политиками маршрутизации. Эта технология называется Policy Based Routing – маршрутизация на основе политик. Суть её в том, что основываясь на каких-либо критериях сетевого пакета, мы либо выбираем таблицу, в которой надо искать маршрут, либо определяем действие, которое надо выполнить над пакетом. Каждая политика имеет номер (он может быть даже не уникальным), он же определяет приоритет. Просмотр политик осуществляется в порядке возрастания их приоритетов. Новые политики добавляются перед существующими.

На данный момент «критериями» политики являются:

– адреса источника и/или назначения;

– значение поля tos;

– интерфейс, с которого получен пакет;

–интерфейс, к которому привязан сокет;

– метка файерволла.

Каждая политика имеет тип, который определяет действие над пакетом, если он под неё попадает:

unicast – используется по-умолчанию, при этом будет производиться поиск

маршрута в заданной таблице маршрутизации;

blackhole/prohibit/unreachable – по действиям аналогичны соответствующим типам маршрутов;

nat – трансляция адресов без учёта состояний практически не используется.

Задание

Создайте новый маршрут для вашего компьютера и проследите его.

1 Запустите виртуальную машину и загрузите ОС Windows.

2 Откройте консоль (Пуск/Программы/Стандартные/Командная строка).

3 Определите IP-адрес вашего компьютера с помощью утилиты ipconfig.

4 Просмотрите таблицу маршрутизации на вашем компьютере: выведите справку по команде route (для этого необходимо ввести команду и нажать клавишу ENTER); выведите таблицу маршрутизации командой route с параметром PRINT: route PRINT; запомните маршрут по умолчанию (первая строка).

5 Проследите работу маршрутизатора с помощью утилиты TRACERT, отправив пакеты на узел www.opennet.ru. Введите: tracert www.opennet.ru.

6 Добавьте в таблицу маршрутизации компьютера строку для пересылки пакетов в сеть 172.21.0.0 (маска 255.255.0.0) через сетевой интерфейс компьютера. Введите: route add 172.21.0.0 mask 255.255.0.0 192.168.1.4 METRIC 3.

7 Проверьте работу внесенных вами изменений с помощью утилиты TRACERT.

Контрольные вопросы

1 Назовите типы маршрутов.

2 Что такое таблица маршрутизации?

3 Что такое метрика в таблице маршрутизации?

4 Что такое шлюз?

5 Что такое адрес шлюза?

11 Лабораторная работа № 11. Обеспечение доступа в сеть Интернет

Цель работы: изучить различные варианты подключения к сети Интернет через локальную сеть, используя программные средства.

Постановка задачи. Имеется локальная сеть (Workstantion 1 – Workstantion 2), представленная на рисунке 2. На компьютере (шлюзе), через который планируется подключение локальной сети к Интернету, необходимо наличие двух сетевых адаптеров (подключений). Требуется обеспечить доступ к сети Интернет со всех рабочих станций.

Имеются три основных варианта подключения локальной сети к Интернету: «прямое» IP-подключение, подключение через NAT, подключение

через прокси-сервер. Рассмотрим преимущества, недостатки и область применения каждого метода, а также некоторые возникающие нюансы. Выбор конкретного способа подключения зависит от потребностей пользователей, цели подключения и, в некоторой степени, финансовых возможностей.

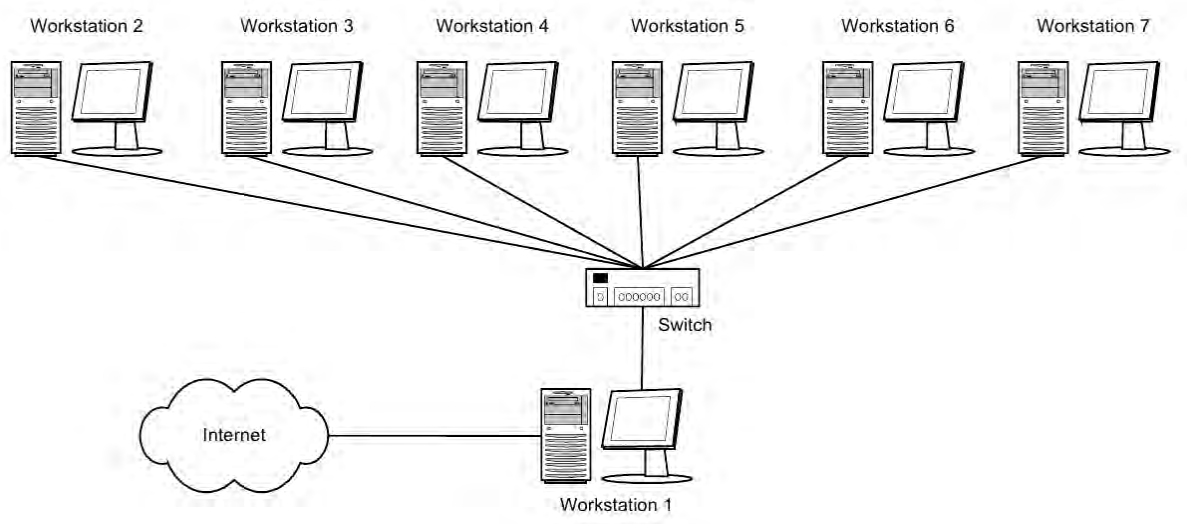

Рисунок 2 – Локальная сеть

Итак, компьютер Workstantion 1. У него есть доступ как к Интернету, так и к локальной сети. Наша задача – дать компьютерам локальной сети доступ к Интернету через подключенный к нему компьютер. Далее этот компьютер мы будем называть шлюзом или маршрутизатором.

Рассмотрение способов мы начнем с наименее часто использующегося, наиболее дорогого, но также наиболее «правильного» и естественного способа, дающего наибольшие по сравнению с другими способами возможности.

«Прямое» IP-подключение к Интернету. Для того, чтобы ваша локальная сеть была полноценно подключена к Интернету, должны соблюдаться, как минимум, три условия: каждая машина в локальной сети должна иметь «реальный», IP-адрес в сети Интернет; эти адреса должны быть не любыми, а выделенными вашим провайдером для вашей локальной сети (скорее всего, это будет подсеть класса C); на компьютере-шлюзе, подключенном к двум сетям – локальной сети и сети провайдера, должна быть организована IP-маршрутизация, т. е. передача пакетов из одной сети в другую. В этом случае Ваша локальная сеть становится как бы частью Интернета. Собственно, это тот способ подключения, которым подключены к Интернету сами Интернет-провайдеры и хостинг-провайдеры. В отличие от обычного подключения, рассчитанного на один компьютер, при таком подключении «под клиента» выделяется не один IP-адрес, а несколько, так называемая «IP-подсеть».

При таком способе подключения вы можете организовать в своей сети сервисы, доступные из Интернета – ведь при данном подключении не только Интернет полностью доступен из вашей сети, но и ваша сеть – из Интернета, т. к. является его частью. Однако такая «прозрачность» вашей сети резко снижает ее защищенность – ведь любые сервисы в локальной сети, даже

предназначенные для «внутреннего» использования, станут доступными извне через Интернет. Чтобы это не имело места, доступ в локальную сеть извне несколько ограничивают. Обычно это делается установкой на шлюзе программы-firewall. Это своеобразный фильтр пакетов, проходящих из одной сети в другую. Путем его настройки можно запретить вход-выход из локальной сети пакетов, соответствующих определенным критериям – типу IP-пакета, IP-адресу назначения, TCP/UDP-порту и т. п.

Firewall решает такие задачи, как:

– блокировку доступа извне к определенным TCP/IP-сервисам локальной сети;

– блокировку доступа к определенным компьютерам локальной сети, таким образом, можно запретить доступ извне ко всем машинам, кроме определенных серверов, предназначенных для доступа в Интернет;

– защиту от троянских программ на сетевом уровне.

Несмотря на универсальность такого метода подключения локальной сети к Интернету, этот метод имеет недостатки. Его используют только лишь те организации, которым надо сделать свои сервера доступными из Интернета – в основном, те же Интернет-провайдеры и хостинг-провайдеры, а также информационные службы. Самый главный недостаток заключается в дороговизне выделения IP-адресов и уж тем более IP-подсетей, к тому же эту плату надо вносить периодически.

Поэтому на практике рассмотрим другие способы, не требующие больших затрат и, что самое главное, позволяющие подключить локальную сеть через обычное подключение с одним внешним IP-адресом.

Подключение через NAT (IP-маскарадинг). Технология Network Address Translation (NAT) – «трансляция сетевых адресов» позволяет нескольким машинам локальной сети иметь доступ к Интернету через одно подключение и один реальный внешний IP-адрес. Для того, чтобы компьютеры локальной сети могли устанавливать соединения с серверами сети Интернет, нужно, чтобы: IP-пакеты, адресованные серверу в Интернете, смогли его достигнуть; ответные IP-пакеты, идущие от сервера Интернета на машину в локальной сети, также смогли ее достигнуть.

С первым условием проблем не возникает, а как быть со вторым? Ведь компьютеры локальной сети не имеют своего «реального» IP-адреса в Интернете! Как же они могут получать IP-пакеты из Интернета?

А работает это следующим образом – на компьютере-шлюзе стоит программа NAT-сервера. Компьютер-шлюз прописан на машинах локальной сети как «основной шлюз», и на него поступают все пакеты, идущие в Интернет (не адресованные самой локальной сети). Перед передачей этих IP-пакетов в Интернет NAT-сервер заменяет в них IP-адрес отправителя на свой, одновременно запоминая у себя, с какой машины локальной сети пришел этот IP-пакет. Когда приходит ответный пакет (на адрес шлюза, конечно), NAT определяет, на какую машину локальной сети его надо направить. Затем в полученном пакете меняется адрес получателя на адрес нужной машины, и пакет доставляется этой машине через локальную сеть.

Как видим, работа NAT-сервера прозрачна для машин локальной сети (как

и работа обычного IP-маршрутизатора). Единственным принципиальным ограничением этого метода подключения локальной сети к Интернету является невозможность установить входящее TCP-соединение из Интернета на машину локальной сети. Однако для «клиентских» сетей этот недостаток превращается в достоинство, резко увеличивающее (по сравнению с первым методом подключения) их защищенность и безопасность. Администраторы некоторых провайдеров даже употребляют слова NAT и Firewall как синонимы.

Подключение через прокси-сервер. Это самый простой тип подключения. При этом никакой маршрутизации IP-пакетов между локальной сетью и сетью Интернет не происходит. Машины локальной сети работают с Интернетом через программу-посредник, так называемый прокси-сервер, установленный на компьютере-шлюзе.

Основной особенностью этого метода является его «непрозрачность». Если, скажем, в случае NAT программа-клиент просто обращается к Интернет-серверу, не «задумываясь», в какой сети и через какую маршрутизацию она работает, то в случае работы через прокси-сервер программа должна явно обращаться к прокси-серверу. Мало того, клиентская программа должна уметь работать через прокси-сервер. Однако проблем с этим не возникает – все браузеры умеют работать через прокси-сервера.

Другой особенностью является то, что прокси-сервер работает на более высоком уровне, чем, скажем, NAT. Здесь уже обмен с Интернетом идет не на уровне маршрутизации пакетов, а на уровне работы по конктретным прикладным протоколам (HTTP, FTP, POP3...). Соответственно для каждого протокола, по которым должны «уметь» работать машины локальной сети, на шлюзе должен работать свой прокси-сервер.

Эта «протокольная зависимость» и есть основной недостаток этого метода подключения как самостоятельного. Однако, с другой стороны, «маршрутизация» на таком высоком уровне может дать и немалые преимущества.

Почти каждый Интернет-провайдер имеет один или несколько прокси-серверов, через которые рекомендует работать своим клиентам. Несмотря на то, что это совершенно необязательно (как правило, клиент провайдера может обращаться к Интернету напрямую), это дает выигрыш в производительности, а при повременной оплате, соответственно, экономить время он-лайн. Это происходит потому, что прокси-сервера способны кэшировать (запоминать) запрашиваемые пользователем документы, и при следующих к ним обращениях выдавать копию из кэша, что быстрее, чем повторно запрашивать с Интернет-сервера. Кроме того, прокси-сервера могут быть настроены так, что будут блокировать загрузку баннеров наиболее распространенных баннерных служб, тем самым также (порой значительно) ускоряя загрузку веб-страниц.

При установке HTTP прокси-сервера в локальной сети и работе через него за счет кэширования экономится не только время, но и трафик – потому что кэширование происходит в самой локальной сети, «до» канала с провайдером, в котором считается трафик (при оплате за объем перекачанной информации).

Задание

На виртуальных машинах осуществите все три способа подключения к Интернету («Прямое» IP-подключение к Интернету, подключение через NAT и подключение через прокси-сервер).

Контрольные вопросы

1 Назовите способы подключения к Интернету.
2 Опишите способ подключения к Интернету через NAT.
3 Опишите способ подключения к Интернету через прокси-сервер.
4 Опишите способ подключения к Интернету через Internet Connection Sharing.

12 Требования к отчетам и защите лабораторных работ

Лабораторная работа направляется на доработку, если количество ошибок и погрешностей позволяет отнести её к низкому уровню соответствия. Допустимые погрешности и ошибки при определении учебных достижений представлены в таблице 2.

Таблица 2 – Допустимые погрешности и ошибки при определении учебных достижений студентов

Шкала соответствия	Уровень соответствия	Балл	Количество ошибок, погрешности / несущественные / существенные
Соответствие	Высокий	5	3/2/0
	Средний	4	6/3/2
	Минимально необходимый	3	7/4/3
Несоответствие	Низкий	2	8/5/4

Погрешностями при определении учебных достижений считаются:
– неточные выражения в отчете по лабораторной работе;
– нерациональные, но правильные приемы, используемые для решения поставленных задач;
– незначительные погрешности при определении параметров.
К несущественным ошибкам относятся:
– неточности определения характеристик и параметров;
– нерациональный способ решения задачи или план ответа (нарушение логики изложения материала, подмена основных понятий второстепенными);
– несоблюдение требований ГОСТа и небрежное оформление отчета по лабораторной работе и графического материала.
К существенным ошибкам относятся:
– подмена понятий в изложении основных понятий;

– незнание фундаментальных понятий серверных операционных систем;

– неумение создавать учетные записи и диагностировать серверные операционные системы;

– неумение в ответе объяснить материал, делать выводы и обобщения, неумение письменно оформить материал;

– незнание клиентских операционных систем.

Оформление рисунков и схем в отчете по лабораторным работам должно соответствовать требованиям ГОСТ 2.105–95. Текстовая часть отчета выполняется либо чертежным шрифтом по ГОСТ 2304–81 с высотой букв не менее 5 мм либо машинным способом шрифтом Times с высотой букв 14 пунктов через одинарный интервал.

Формулы, иллюстрации и таблицы нумеруются в пределах отчета. Обозначения переменных и параметров, принятых в формулах, должны быть расшифрованы сразу после написания формулы. При этом указываются единицы измерения переменных и параметров.

Рисунки, графики и таблицы сопровождаются наименованиями, отображающими их содержание (например, Рисунок 1 – Физическая топология компьютерной сети). Если на одном рисунке изображено несколько графиков различных процессов, то каждый график должен иметь отдельное обозначение, которое необходимо расшифровать в поясняющих данных к рисунку. Поясняющие данные помещаются под рисунком перед его наименованием.

Отчет или его часть могут быть представлены в электронном виде по согласованию с преподавателем.

Общие требования к содержанию отчета

1 Тема, цель работы.

2 Постановка задачи.

3 Вариант задания с исходными данными.

4 Выполненное задание согласно варианту: алгоритм решения поставленной задачи; результаты выполненного задания.

5 Результаты тестирования задания.

6 Выводы по теме лабораторной работы.

Список литературы

1 Microsoft Windows Server 2012. Полное руководство / Р. Моримото [и др.]. – Москва : Вильямс, 2013. – 1456 с.

2 **Айвалиотис, Д. И.** Администрирование сервера NGINX : учебник / Д. И. Айвалиотис. – Москва : ДМК Пресс, 2013. – 1275 с.

3 **Бройдо, О. П.** Вычислительные системы, сети и телекоммуникации : учебник / О. П. Бройдо, В. Л. Бройдо, О. П. Ильина. – 4-е изд. – Санкт-Петербург : Питер, 2011. – 560 с.

4 **Варфоломеева, А. О.** Информационные системы предприятия : учебное пособие / А. О. Варфоломеева, А. В. Коряковский, В. П. Романов. – Москва : ИНФРА-М, 2013. – 283 с.

5 **Васильков, А. В.** Информационные системы и их безопасность : учебное пособие / А. В. Васильков, А. А. Васильков, И. А. Васильков. – Москва : Форум, 2013. – 528 с.

6 **Гвоздева, В. А.** Информатика, автоматизированные информационные технологии и системы : учебник / В. А. Гвоздева. – Москва : ФОРУМ ; ИНФРА-М, 2013. – 544 с.

7 **Дворкович, В. П.** Цифровые видеоинформационные системы (теория и практика) / В. П. Дворкович, А. В. Дворкович. – Москва : Техносфера, 2012. – 1008 с.

8 **Емельянов, С. В.** Информационные технологии и вычислительные системы. Интернет-технологии. Математическое моделирование. Системы управления. Компьютерная графика / С. В. Емельянов. – Москва : Ленанд, 2012. – 96 с.

9 **Емельянов, С. В.** Информационные технологии и вычислительные системы. Вычислительные системы. Математическое моделирование. Прикладные аспекты информатики / С. В. Емельянов. – Москва : Ленанд, 2015. – 96 с.

10 **Епашников, А. П.** Локальные вычислительные сети : учебник / А. П. Епашников, В. А. Епашников. – Москва : Диалог-МИФИ, 2013. – 380 с.

11 Информационные системы и технологии управления : учебник / Под ред. Г. А. Титоренко. – Москва : ЮНИТИ, 2013. – 591 с.

12 Информационные системы и технологии / Под ред. Ю. Ф. Тельнова. – Москва : ЮНИТИ, 2016. – 303 с.

13 **Кенин, А.** Самоучитель системного администратора / А. Кенин. – Санкт-Петербург : БХВ-Петербург, 2012. – 512 с.

14 **Миков, А. И.** Информационные процессы и нормативные системы в IT: Математические модели. Проблемы проектирования. Новые подходы / А. И. Миков. – Москва : КД Либроком, 2013. – 256 с.

15 **Новиков, В. А.** Информационные системы и сети : учебное пособие / В. А. Новиков, А. В. Новиков, В. В. Матвеенко. – Минск : Изд-во Гревцова, 2014. – 448 с.

16 **Олейник, П. П.** Корпоративные информационные системы : учебник для вузов / П. П. Олейник. – Санкт-Петербург : Питер, 2012. – 176 с.

17 **Олифер, В. Г.** Компьютерные сети. Принципы, технологии, протоколы :

учебное пособие / В. Г. Олифер, Н. А. Олифер. – 4-е изд. – Санкт-Петербург : Питер, 2013. – 944 с. : ил.

18 **Поляк-Брагинский, А.** Администрирование сети на примерах / А. Поляк-Брагинский. – Санкт-Петербург : БХВ-Петербург, 2012. – 432 с.

19 **Сырецкий, Г. А.** Информатика. Фундаментальный курс. Т. 2 : Информационные технологии и системы / Г. А. Сырецкий. – Санкт-Петербург : BHV, 2012. – 848 с.

20 **Федорова, Г. Н.** Информационные системы: учебник / Г. Н. Федорова. – Москва : Академия, 2013. – 208 с.

21 **Федотова, Е. Л.** Информационные технологии и системы : учебное пособие / Е. Л. Федотова. – Москва : ФОРУМ ; ИНФРА-М, 2013. – 352 с.