В ходе выполнения лабораторной работы студент должен выполнить предложенное задание и подготовить отчет о проделанной работе. Форма сдачи лабораторной работы предполагает демонстрацию выполненного задания и знаний теоретической части вопроса, рассмотренного в лабораторной работе.

СОДЕРЖАНИЕ

[Лабораторная работа 1 Программное обеспечение для организации «виртуальных машин» - VMware Player. 3](#_Toc492382268)

[Лабораторная работа 2 Установка FreeBSD 11.1 на «виртуальную машину» на базе VMware Player. 15](#_Toc492382269)

[Лабораторная работа 3 Работа в ОС FreeBSD. Терминалы, текстовый режим работы. Простые команды. Процессы. 32](#_Toc492382270)

[Лабораторная работа 4 Работа в ОС FreeBSD. Файловая система. Управление каталогами. 40](#_Toc492382271)

[Лабораторная работа 5 Работа в ОС FreeBSD.Пользователи, группы. Управление правами доступа. 50](#_Toc492382272)

[Лабораторная работа 6 Первоначальная настройка сервера на основе FreeBSD 11.1. Текстовый редактор «ee». Команда su. Утилита sudo. Установка bash. Локализация консоли. Настройка ssh, ftp. 56](#_Toc492382273)

[Лабораторная работа 7 Настройка Web-сервера (Apache). Организация файлового сервера на основе Samba. 66](#_Toc492382274)

[Лабораторная работа 8 Работа в ОС FreeBSD. Обработка текста: grep, sed, awk. 73](#_Toc492382275)

[Лабораторная работа 10 Планирование адресного пространства в IP сетях. 82](#_Toc492382276)

[Лабораторная работа 11 Знакомство с программами Boson Network Designer и NetSim. 97](#_Toc492382277)

[Лабораторная работа 12 Базовая конфигурация маршрутизатора 107](#_Toc492382278)

[Лабораторная работа 13 Статическая маршрутизация. 114](#_Toc492382279)

[Лабораторная работа 14 Динамическая маршрутизация (протокол RIP). 116](#_Toc492382280)

[Лабораторная работа 15 Комплексная работа по настройке сети. 118](#_Toc492382281)

# Лабораторная работа 1 Программное обеспечение для организации «виртуальных машин» - VMware Player.

**Цель**

Изучить основные возможности VMware Player. Научиться настраивать параметры оборудования виртуальной машины.

**Теоретические сведения**

Виртуализация обеспечивает выполнение нескольких виртуальных машин на одном физическом компьютере и совместное использование его ресурсов виртуальными машинами в нескольких средах. Различные виртуальные машины могут использовать разные ОС. Кроме того, на одном компьютере можно выполнять несколько приложений.

Виртуализация ИТ-инфраструктуры обеспечивает снижение расходов на ИТ и при этом увеличивает эффективность, коэффициент использования и повышает гибкость имеющихся активов.

**5 основных причин внедрить ПО виртуализации**

1. **Увеличение отдачи от существующих ресурсов**: объединение общих ресурсов инфраструктуры в пулы и уход от устаревшей модели «один сервер — одно приложение» с помощью [консолидации серверов](http://www.vmware.com/ru/solutions/datacenter/consolidation/index.html).
2. **Снижение расходов на ЦОД за счет уменьшения физической инфраструктуры и увеличения числа серверов на число администраторов**: уменьшение числа серверов и сопутствующих устройств влечет за собой уменьшение необходимой площади помещений и [сокращение потребностей в электроэнергии на питание и охлаждение](http://www.vmware.com/ru/virtualization/green-it/index.html). Более эффективные средства управления увеличивают число серверов на одного администратора. Следовательно, снижается потребность в кадрах.
3. **Увеличение доступности оборудования и** [приложений](http://www.vmware.com/ru/solutions/business-critical-apps/index.html) **для повышения уровня** [непрерывности бизнеса](http://www.vmware.com/ru/solutions/datacenter/business-continuity/): надежное резервное копирование и перенос виртуальных сред целиком без прерывания работы. Исключение плановых простоев и быстрое восстановление после непредвиденных сбоев.
4. **Эксплуатационная гибкость**: оперативное реагирование на изменения рынка благодаря динамическому управлению ресурсами, ускоренной инициализации серверов и улучшенному развертыванию настольных компьютеров и приложений.
5. **Улучшение управляемости и безопасности настольных компьютеров**: развертывание, администрирование и мониторинг [безопасных сред настольных компьютеров](http://www.vmware.com/ru/solutions/desktop/), к которым пользователи могут обращаться локально или удаленно через сетевое подключение или без него, используя практически любой стандартный настольный компьютер, ноутбук или планшетный ПК.

**Виртуальная машина** — это полностью изолированный программный контейнер, который работает с собственной ОС и приложениями, подобно физическому компьютеру. Виртуальная машина работает полностью аналогично физическому компьютеру и обладает собственными ЦП, памятью, жестким диском и сетевым адаптером.

Установленная на ней операционная система, а также приложения и другие компьютеры в сети оказываются не в состоянии отличить, имеют ли они дело с виртуальной или физической машиной. Даже сама виртуальная машина считает себя «настоящим» компьютером. Но несмотря на это виртуальные машины состоят исключительно из программных компонентов и не включают оборудование. Это дает им ряд уникальных преимуществ над физическим оборудованием.

**Преимущества виртуальных машин**

В общем и целом, виртуальные машины VMware предлагают четыре основные характеристики, полезные для заказчиков.

**Совместимость**

Подобно физическому компьютеру, виртуальная машина работает с собственной гостевой ОС и приложениями и имеет все необходимые компоненты (материнскую плату, VGA- и сетевой адаптеры и т.п.). Таким образом, виртуальные машины полностью совместимы со всеми стандартными ОС, приложениями и драйверами для платформы x86, поэтому в виртуальной машине можно запускать то же ПО, что на физическом компьютере x86.

**Изоляция**

Виртуальные машины могут совместно использовать ресурсы одного компьютера, но при этом они сохраняют полную изоляцию друг от друга, как если бы они были отдельными физическими компьютерами. Например, если на одном физическом сервере работает четыре виртуальные машины и одна из них отказывает, остальные три остаются доступны. Изоляция — это одна из основных причин значительно более высокой доступности и безопасности приложений в виртуальной среде по сравнению с традиционной невиртуализированной системой.

**Инкапсуляция**

По сути, виртуальная машина — это программный контейнер, который объединяет в себе или «инкапсулирует» полный набор аппаратных ресурсов, а также ОС, приложения и все приложения в программном пакете. Инкапсуляция делает виртуальные машины невероятно мобильными и удобными в управлении. Например, вы можете переносить и копировать виртуальные машины так же, как любой другой программный файл, и сохранять их на любом стандартном устройстве хранения от карманных USB-накопителей до корпоративных хранилищ SAN.

**Независимость от оборудования**

Виртуальные машины полностью независимы от базового физического оборудования. Например, можно настроить виртуальную машину с виртуальными комплектующими (например, ЦП, сетевым адаптером и SCSI-контроллером), полностью отличными от физических комплектующих базовой системы. Более того, виртуальные машины на одном сервере могут работать под управлением разных ОС (Windows, Linux и др).

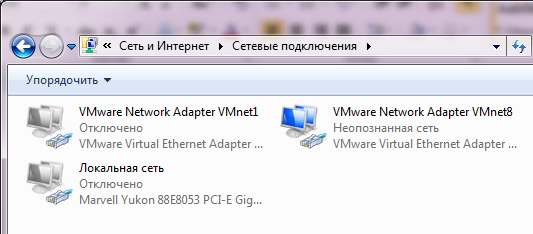
В сочетании с инкапсуляцией и совместимостью независимость от оборудования обеспечивает возможность переноса виртуальной машины с одного типа компьютеров x86 на другой без внесения изменений в драйверы устройств, операционную систему и приложения. Независимость от оборудования означает также поддержку разнородных комбинаций ОС и приложений на одном физическом сервере.

**VMware Player** – это программный продукт, который помогает легко создавать и запускать виртуальные машины на ПК под управлением ОС Windows или Linux. VMware Player поддерживает создание виртуальных машин, а также запуск виртуальных машин, созданных с помощью [VMware Workstation](http://www.vmware.com/ru/products/desktop_virtualization/workstation/overview.html), [VMware Fusion](http://www.vmware.com/ru/products/desktop_virtualization/fusion/overview.html), VMware Fusion Professional и[VMware vSphere](dcr://templatedata/content/standard/data/pages/products/vsphere). Является бесплатным для некоммерческого использования.

Рассмотрим возможные варианты настройки доступа к сети для виртуальной машины. Для начала определимся, что

* **Физическая сетевая карта** – это сетевая карта, которая присутствует на Вашем компьютере и позволяет Вам получать доступ к локальной сети, так возможно и к Интернет.
* **Эмулятор сетевой карты** – это «программная» сетевая карта, т.е. ее физически не существует. Эта карта работает только в пределах Вашей базовой (хостовой) операционной системы. По умолчанию эмулятор сетевой карты в VMware Player называется «VMware Network Adapter VMnet*X*»(*X* – это номер адаптера, от 0 до 10).
* **Сетевая карта гостевой операционной системы** – это программная эмулируемая карта (не физическая), которая находится внутри гостевой операционной системы.

После установки VMware Player в Вашей базовой (хостовой) операционной системе появятся дополнительные «виртуальные» сетевые адаптеры.

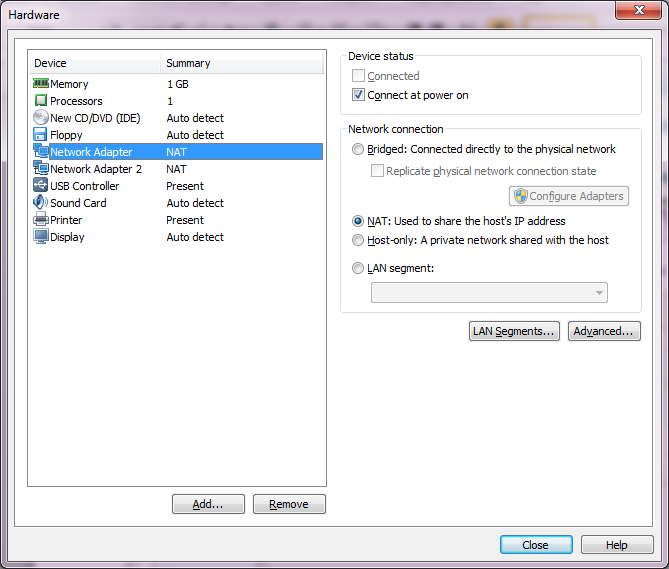


При настройке сетевого адаптера виртуальной машины можно выбрать один из трех вариантов сетевого подключения:

1) **«Bridget»** (Мост). В этом режиме «Физическая сетевая карта» и «Сетевая карта гостевой операционной системы» работают как единый механизм, то есть на физическом хосте не будет задействован «Эмулятор сетевой карты» для этого подключения, и в роли «Эмулятора сетевой карты» выступает «Физическая сетевая карта». При этом режиме виртуальная машина находится в одной сети (сегменте) с сетью, в которой находится базовый (хостовой) компьютер. Реализован как бы виртуальный коммутатор (vmnet0), к которому подключена и физическая сетевая карта, и сетевая карта гостевой операционной системы. В этом случае IP-адреса сетевой карты гостевой ОС должны принадлежать диапазону IP-адресов сети хостового компьютера.

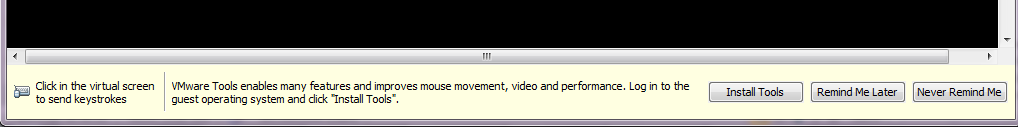
2) **«Nat»** (Network address translation, Трансляция сетевых адресов). В этом режиме «Сетевая карта гостевой операционной системы» будет получать доступ в сеть от «Эмулятора сетевой карты», транслируемого с физического хоста, и выступающего в роли роутера (как если бы это было подключение от провайдера). Во многих случаях этого подключения будет более чем достаточно, например для того что бы можно было оперативно выйти в интернет с гостевой операционной системы. При этом все запросы на гостевой ОС будут происходить от имени физического хоста. При таком соединении IP-адреса виртуальной машины (сетевой карты гостевой ОС) будут отличаться от IP-адресов хостового компьютера. Для такого подключения используется виртуальный коммутатор vmnet8, к которому подключен интерфейс VMware Network Adapter VMnet8. По умолчанию этому адаптеру уже присвоен IP-адрес, но его можно поменять. Для этого типа подключения работает специальная служба, которая была установлена вместе с VMware Player – VMware DHCP. Служба эмулирует работу DHCP сервера, выдающего IP-адреса сетевым интерфейсам виртуальной машины (сетевым картам гостевой ОС), для которых указан тип сетевого подключения NAT. Кроме этого VMware Player устанавливает службу VMware NAT, которая обеспечивает доступ из виртуальных машин к различным интернет-сервисам.

3) **«Host-only»**(Только хост). В этом режиме «Сетевая карта гостевой операционной системы» и «Эмулятор сетевой карты» будут работать как простое сетевое соединение между двумя компьютерами, например как два физических компьютера соединенных витой парой в режиме «кроссовер». Для данного подключения используется виртуальный коммутатор vmnet1, к которому подключен интерфейс VMware Network Adapter VMnet1. В этой «виртуальной сети» так же работает VMware DHCP-сервер. Данное подключение актуально, если, например виртуальной машине (гостевой операционной системе), требуется перекрыть доступ к интернету (создать приватную сеть без интернета), хотя если постараться и настроить маршрутизацию…

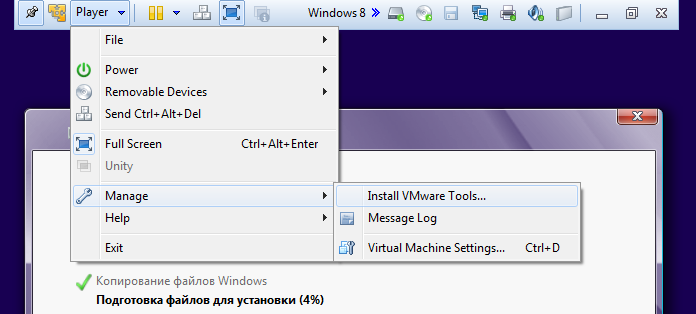


Опция «LAN Segments» используется, если нужно не просто построить сеть для экспериментов, а например, проэмулировать существующую сеть или несколько ее сегментов не связанных между собой, или создать приватную сеть без подключения виртуальных сетевых адаптеров хостовой операционной системы. Таким образом, разделение подключенных виртуальных машин, возможно, провести благодаря именно этой опции.

Для полноценной работы некоторого оборудования и получения дополнительных возможностей от VMware Player необходимо установить VMware Tools. Программа напомнит Вам об этом при запуске виртуальной машины.



Лучше всего это делать, когда гостевая операционная система установлена и минимально настроена. Установка производится при включенной виртуальной машине. Причем версия VMware Tools зависит не только от ОС, но и ее версии. Для ОС типа Windows установка VMware Tools проходит практически в автоматическом режиме и не вызывает затруднений. Для OC тип Unix приходится намного поработать ручками.



**Задание**

**Замечания (важные ☺):**

1. В качестве платформы виртуализации можно использовать Microsoft  Hyper-V или Oracle VM VirtualBox.
2. Созданая виртуальная машина будет использована в следующей ЛР для установки ОС FreeBSD.

Создать виртуальную машину со следующими параметрами:

* Имя – FreeBSD\_Srv (будем делать сервер на базе FreeBSD 11)
* Оперативная память – не менее 512Мб (лучше 1024 Мб)
* Жесткий диск – первый – не менее 10Гб, второй – не менее 5Гб
* Сетевое подключении – NAT (если в последствии возниктун проблемы с сетевым подключением, то можно попробовать использовать Host-only тип подключения)
* Остальное оборудование оставить по умолчанию (лишнее, например, звуковая карта, можно удалить)

# Лабораторная работа 2 Установка FreeBSD 11.1 на «виртуальную машину» на базе VMware Player.

**Цель**

Изучить процесс установки FreeBSd 11.1: возможные варианты, основные шаги.

**Теоретические сведения**

FreeBSD имеет текстовый установщик. Начиная с FreeBSD 9.0-RELEASE используют установщик **bsdinstall** и **bsdconfig** – для постустановочной настройки, в предыдущих релизах использовался **sysinstall**.

Необходимые системные требования:

FreeBSD/i386 требует 486 или более новый процессор и более 64 MB RAM. Для минимальной установки необходимо не менее 1,1 Gb на HDD.

FreeBSD/amd64 поддерживает два семейства процессоров. Первое семейство это процессоры AMD64 (AMD Athlon™64, AMD Athlon64-FX, AMD Opteron™ и более новые), второе - процессоры Intel® EM64T (Intel Core™ 2 Duo(Quad, Extreme), Intel Xeon™, Intel Core i3, i5, i7).

Перед установкой необходимо провести некоторые подготовительные работы.

Жесткий диск винчестера может быть разбит на несколько секций, которые называют разделами. Существует два способа разбивки. Первый, он же традиционный Master Boot Record (MBR) подразумевает создание 4-х первичных разделов (по историческим причинам FreeBSD называет эти разделы слайсами). В настоящее время, с учетом новых дисков большого размера ограничение в 4 слайса, несмотря на то, что один из них можно определить как расширенный (и создать логические разделы) является неприемлемым. Данное ограничение решает новым методом разбивки - GUID Partition Table (GPT), позволяющим создавать до 128 разделов, избавляя от необходимости создания логических разделов.

Следует учесть, что старые операционные системы, такие как Windows® XP не поддерживают GPT. При совместной установкe следует использовать MBR.

Для стандартного загрузчика FreeBSD 11.1 необходимо наличие свободного первичного раздела MBR или свободного раздела GPT. Для минимальной установки необходимо 1Gb свободного места, но на самом деле для комфортной работы нужно 3Gb без графической оболочки и 5 Gb с графической оболочкой. Установка дополнительных приложений соответственно требует больше места.

Также перед установкой необходимо собрать информацию касательно вашего сетевого окружения, если сетевые настройки вы не получаете по DHCP.

В частности:

* IP адрес (IP address)
* Маску подсети (Subnet mask)
* Шлюз по умолчанию (Default router IP address)
* Доменное имя локальной сети (domain name of the local network)
* Адреса DNS-серверов (DNS server IP address(es))

И самое главное, какую архитектуру имеет процессор на вашем компьютере. Если это 32-х разрядный Intel, значит архитектура – i386. Если процессор 64-х разрядный, то он имеет архитектуру либо [amd64](http://en.wikipedia.org/wiki/X86-64#AMD64), либо [ia64](http://en.wikipedia.org/wiki/X86-64#Intel_64). В случае с виртуальной машиной считаем, что у нас архитектура i386.

Установку FreeBSD можно провести:

* с CD/DVD-дисков, качаем один из образов:
  + <ftp://ftp.freebsd.org/pub/FreeBSD/releases/ISO-IMAGES/11.1/FreeBSD-11.1-RELEASE-i386-bootonly.iso> (позволяет выполнить загрузку, но при установке все необходимое установщик скачивает из интернета)
  + <ftp://ftp.freebsd.org/pub/FreeBSD/releases/ISO-IMAGES/11.1/FreeBSD-11.1-RELEASE-i386-disc1.iso>
  + <ftp://ftp.freebsd.org/pub/FreeBSD/releases/ISO-IMAGES/11.1/FreeBSD-11.1-RELEASE-i386-dvd1.iso>
* с USB-носителя – <ftp://ftp.freebsd.org/pub/FreeBSD/releases/ISO-IMAGES/11.1/FreeBSD-11.1-RELEASE-i386-memstick.img>
* и из сети.

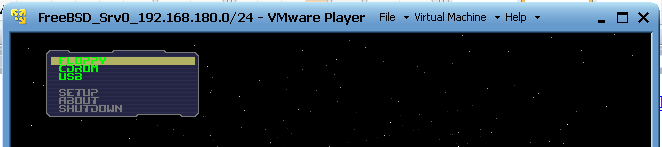
Для записи образа в системе FreeBSD следует ввести команду

***# dd if=FreeBSD-11.1-RELEASE-i386-memstick.img of=/dev/da0 bs=64k***

где подразумевается что /dev/da0 - носитель USB.

В Windows записать образ можно программой [Image Writer for Windows](http://unix.uz/go?a%3AaHR0cHM6Ly9sYXVuY2hwYWQubmV0L3dpbjMyLWltYWdlLXdyaXRlci8%3D). Следует заметить, что данная процедура стирает данные на USB-носителе.

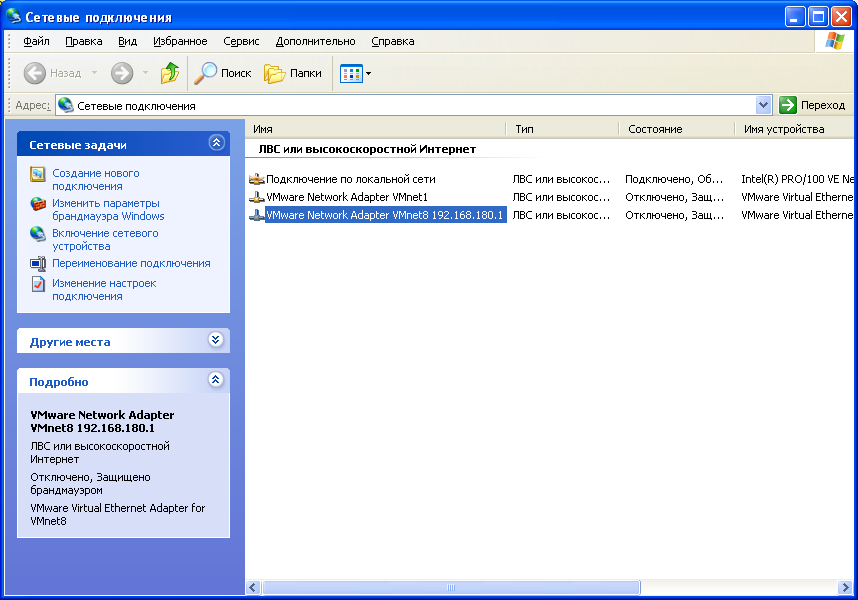
По умолчанию загрузка в VMware Player c загрузочной флешки не возможна. Но при использовании некоторых хитростей можно заставить загружать с флешки. Для загрузки с флешки понадобиться Plop Boot Manager (http://www.plop.at/en/bootmanager.html). Скачиваем, распаковываем в каталог и настройках cd/dvd привода виртуальной машины указываем образ **plpbt.iso**. При загрузке с этого образа получим следующую картинку и выбрав пункт меню USB, сможем загрузиться (начать установку) с USB носителя, на который предварительно записали образ FreeBSD-11.1-RELEASE-i386-memstick.img.



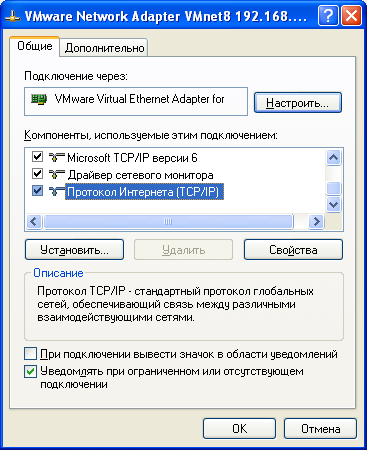
При выполнении лабораторной работы Вы можете воспользоваться любым из способов установки. Только помните, что при установке с FreeBSD-11.1-RELEASE-i386-bootonly.iso необходим доступ в Интернет.

Итак, виртуальная машина настроена (Вы воспользуетесь виртуальной машиной, созданной в предыдущей работе), осталось только немного «допилить»: настроим параметры виртуального интерфейса для обеспечения доступа виртуальной машины к сети. Так как будет использоваться механизм NAT, то необходимо настроить параметры VMware Network Adapter VMnet8 средствами базовой ОС (в случае исползования Host-only типа подключения настраиваем параметры адаптера VMnet1).

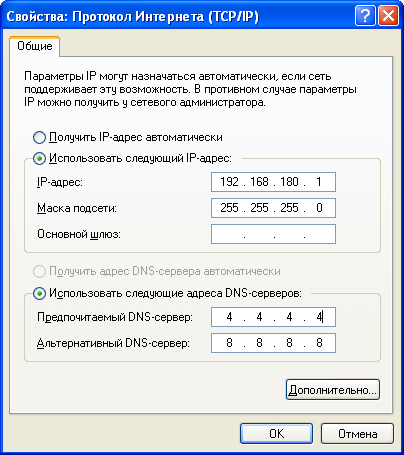
Откроем окно «Сетевые подключения» и найдем виртуальный адаптер VMware Network Adapter VMnet8 (на картинке он уже переименован немного для удобства).



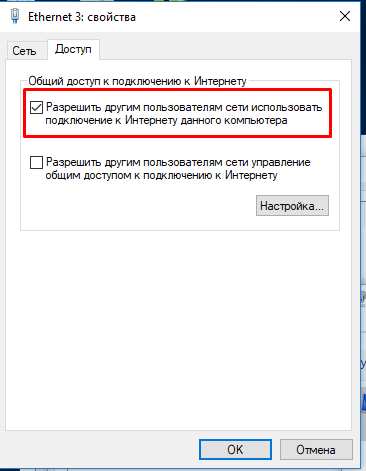
Открываем окно свойств этого адаптера и выбираем компонент «Протокол Интернета (TCP/IP)», жмем кнопку «Свойства».



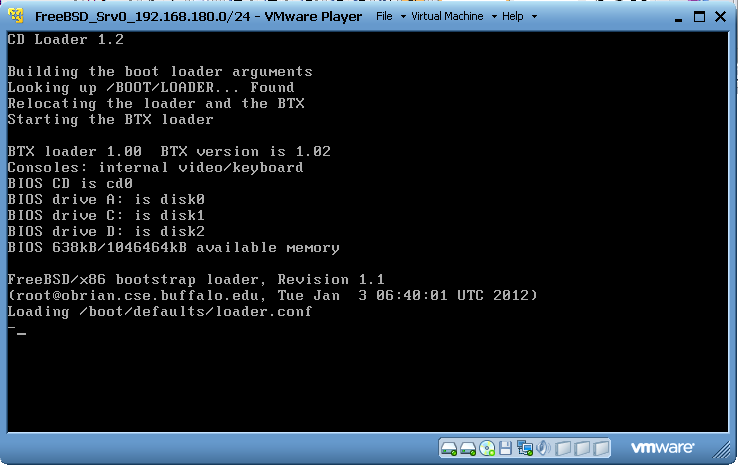
В окне свойств определяем основные настройки сетевого соединения.



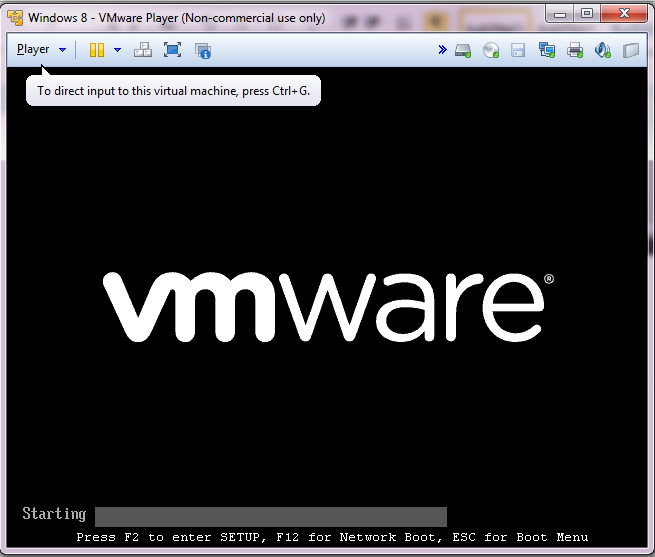
Для полуения доступа к Интернет через хостовую ОС необходимо «расшарить» сетевое подключение на «хостовой» машине. Для этого на вкладке «Доступ» параметров физического сетевого адаптера Вашего компьютера укажите какому виртуальному адаптеру разрешено использовать данное сетевое подключение (если включен только один виртуальный адаптер, то достаточно установить «галочку»).

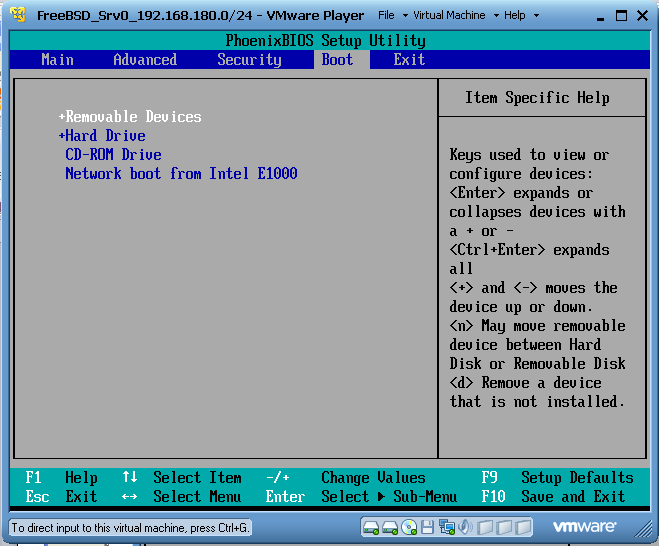


Теперь можно включать виртаульную машину. Так как на виртуальной машине еще не установлена операционная система, то загрузка будет произведена с CD/DVD привода.

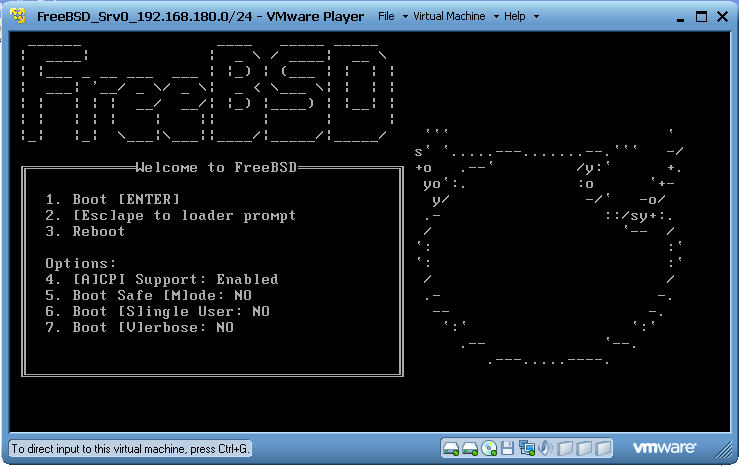


Последовательность устройств, с которых будет пытаться загрузиться виртуальная машина можно настроить в BIOS (как и на обычном компьютере). Для этого при включении виртуальной машины нужно успеть нажать F2 и получим следующий интерфейс.





Получаем меню загрузки, которое пропадет после 10 секунд и загрузка продолжится. Если нажать Enter, то ждать не придется, а если Пробел, то можно будет остановить отсчет времени и выбрать дополнительные опции загрузки.



Далее выполняется процесс определения устройств выводится запрос о дальнейших действиях:

**Install** – установка системы;

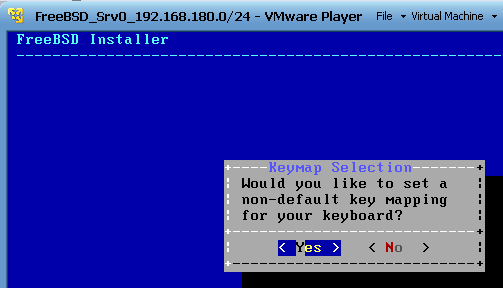
**Shell** – запуск командной оболочки;

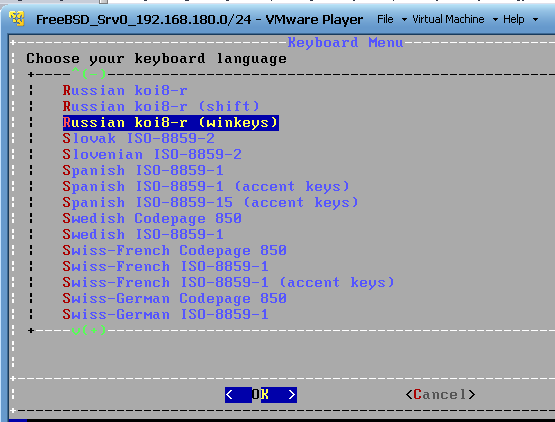
**Live CD** – можно сказать, что это аналог Frenzy.

Frenzy – это «портативный инструмент системного администратора», LiveCD на базе ОС FreeBSD, загрузившись с которого, администратор получает полностью работоспособную систему с набором программного обеспечения для настройки, проверки и анализа сети, тестирования компьютерного «железа» и ряда других задач.

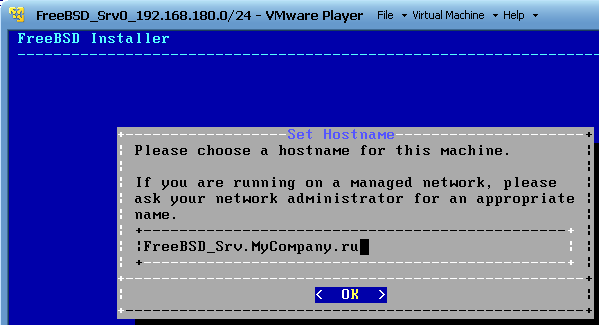


Утвердительно отвечаем на вопрос bsdinstaller об изменении раскладки клавиатуры и выбираем нужную из списка.



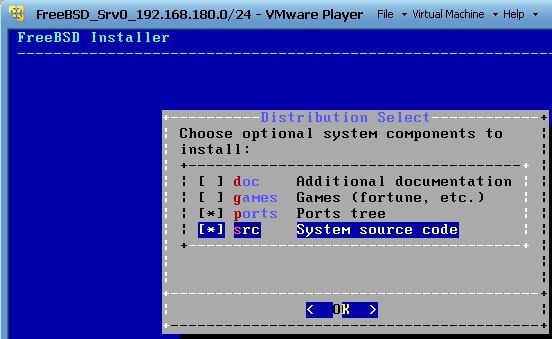


Вводим имя для системы – FreeBSD\_Srv.MyCompany.ru



Выбираем компоненты для установки

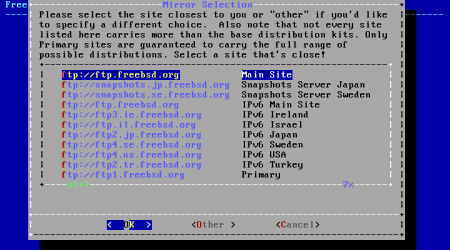
* doc - сопутствующая документация, представляющая в основном исторический интерес;
* games - традиционные игры (**снимаем отметку - пробелом**);
* lib32 - библиотеки, необходимые для запуска 32-х битных приложений на 64-х битной системе (т.к. мы ставим 32-х битную версию, то не видим этого пункта);
* ports - коллекция портов;
* src - исходники системы (**ставим отметку пробелом**).



В случае **bootonly** установки, необходимо наличие настроенного сетевого соединения (желательно высокоскоростного), поскольку данный тип установки не подразумевает наличие устанавливаемых файлов на носителе.

[](http://unix.uz/uploads/posts/2012-02/1329194733_bsdinstall-netinstall-files.png)

После настройки сетевого соединения, о котором будет более подробно рассказано ниже, вам будет необходимо выбрать предпочтительное зеркало и установка будет продолжена после получения установочных файлов на ваш компьютер.

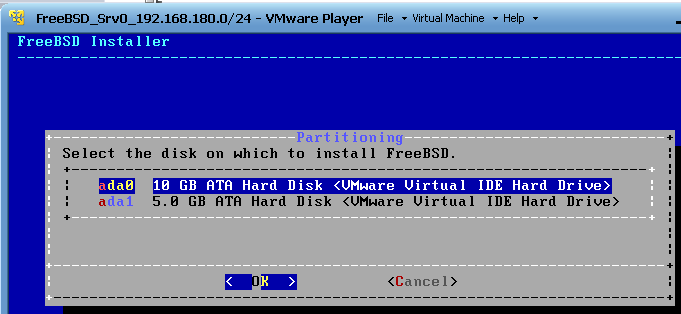
[](http://unix.uz/uploads/posts/2012-02/1329194872_bsdinstall-netinstall-mirrorselect.png)

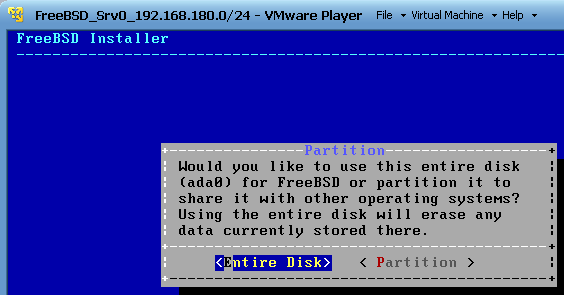
Приступаем к разбиению жесткого диска. Предлагается выбрать из трех вариантов.

* **Guided** - автоматически разбивает диск на разделы.
* **Manual** - позволяет вручную отредактировать разделы.
* **Shell** - запускает командую оболочку для использования таких команд как gpart, fdisk, and bsdlabel.

[](http://unix.uz/uploads/posts/2012-02/1329195144_bsdinstall-part-guided-manual.png)

Выбираем **автоматический** режим и смотрим, что посоветует установщик.

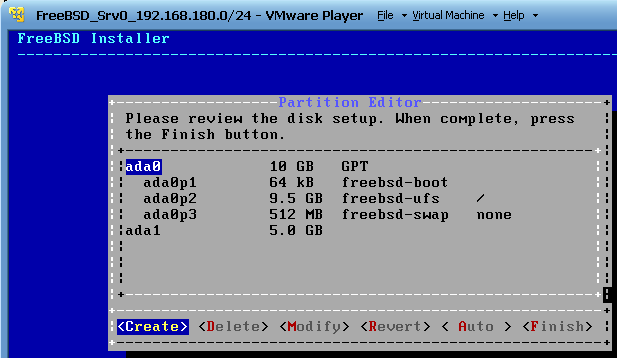




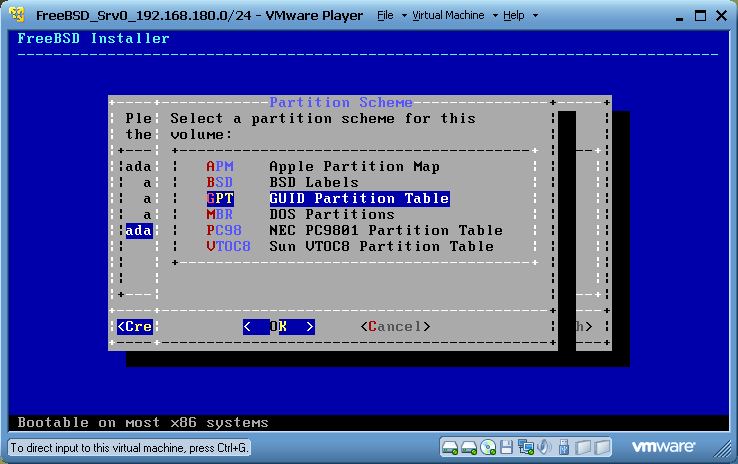
Установщик разбил выбранный диск следующим образом

* **freebsd-boot** - этот раздел должен быть первым. Содержит загрузчик.
* **freebsd-ufs** - раздел UFS FreeBSD.
* **freebsd-swap** - раздел подкачки (swap).

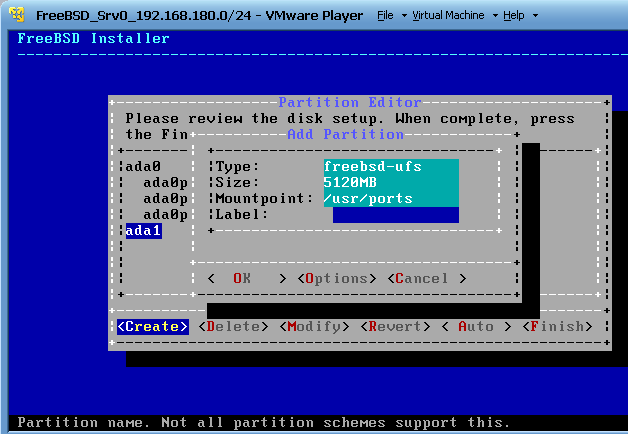
Для указания размера используются общепринятые сокращения: K для килобайтов, M для мегабайтов, или G для гигабайтов.



Самостоятельно размечаем второй диск (**ada1**). Для этого выбираем его и жмем **Create.**

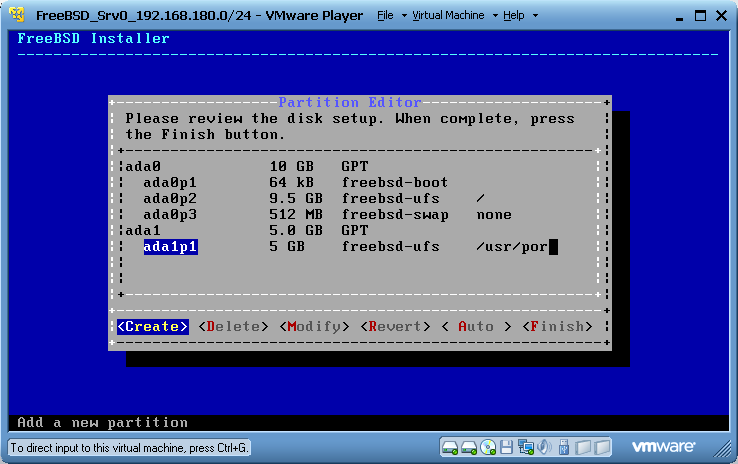


Выбрали метод разбивки – GPT.



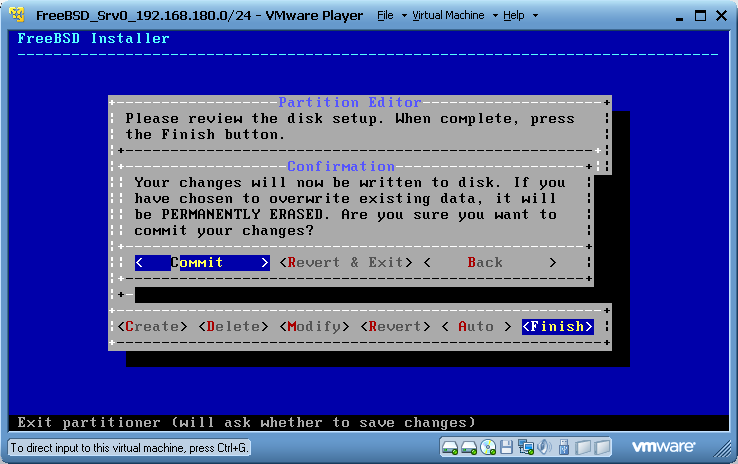
Создаем один раздел и полностью отдаем его под порты (указываем точку монтирования).

Получаем следующий результат.

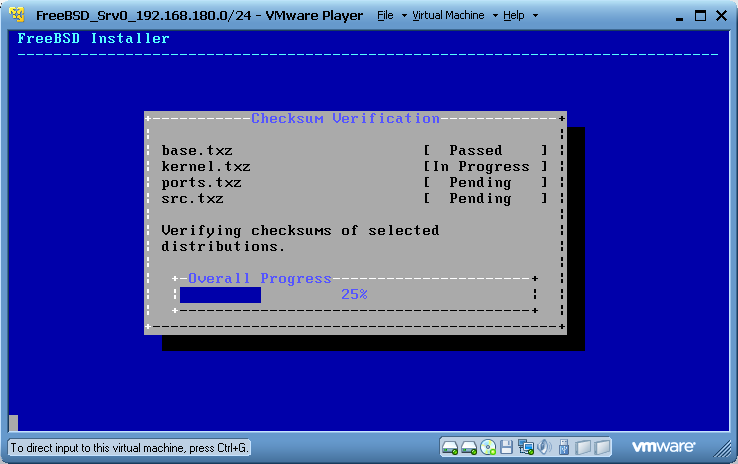


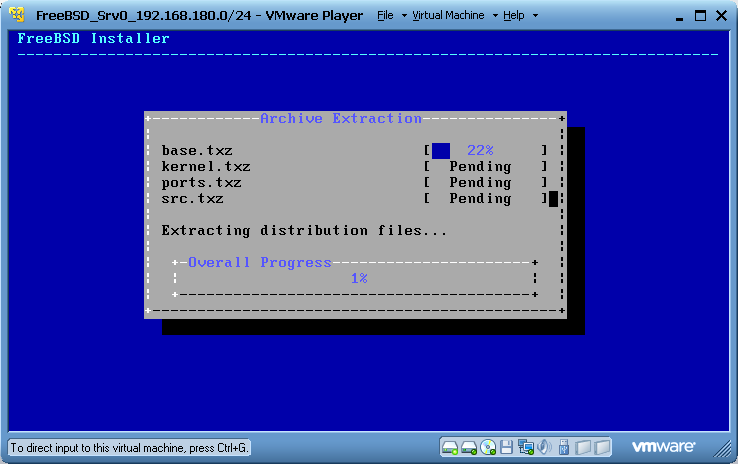
На этом манипуляции с дисками закончили и выбираем **Finish**.

Подтверждаем изменения. Переходим т.н. «точку невозвращения».

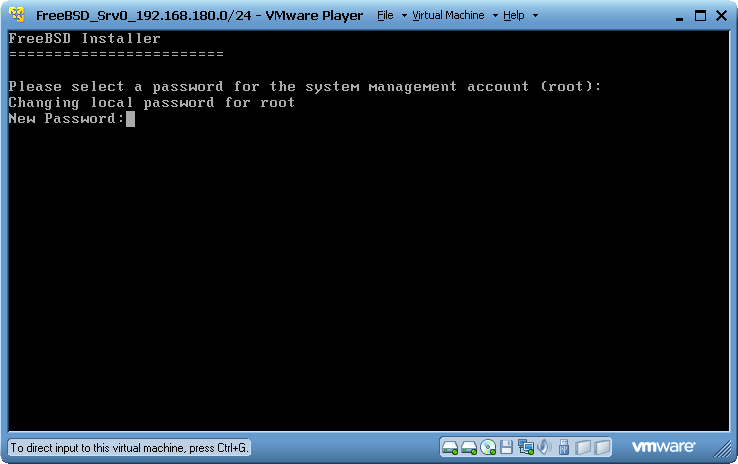


Далее происходит получение установочных файлов по сети, если режим **bootonly**, проверка контрольных сумм и разархивирование.



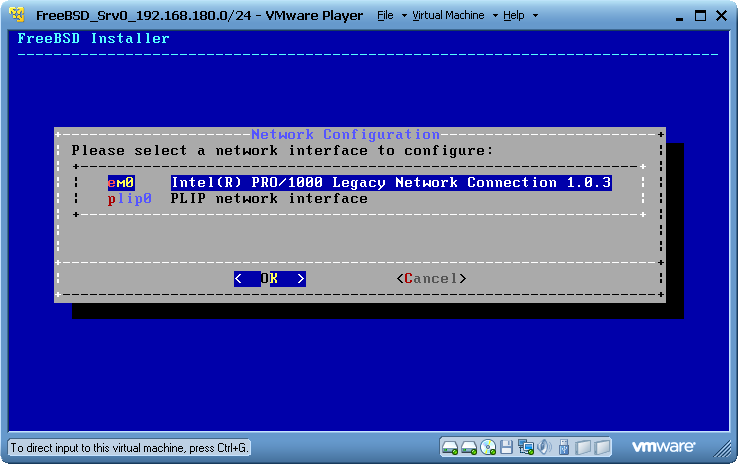


После распаковки **bsdinstall** перейдет к пост-установочной настройке. Укажем пароль для «главного» пользователя – root.



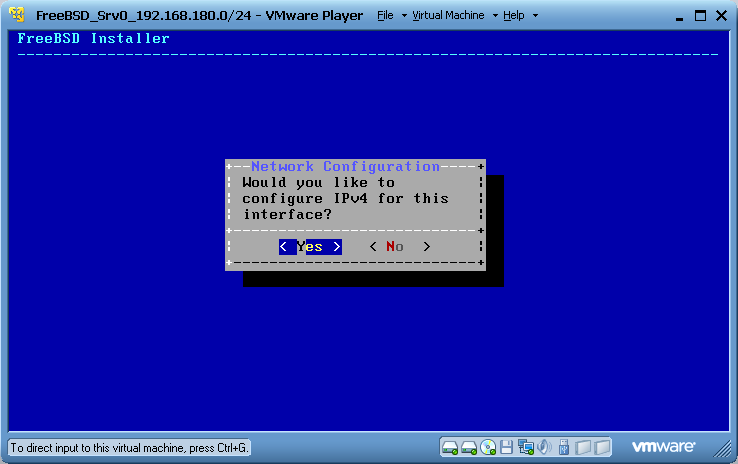
Далее настроим сетевые интерфейсы. (Если они настраивались при bootonly-режиме, то данная часть будет пропущена).

В случае обычной Ethernet сетевой карты все просто, выбираем ее в списке.

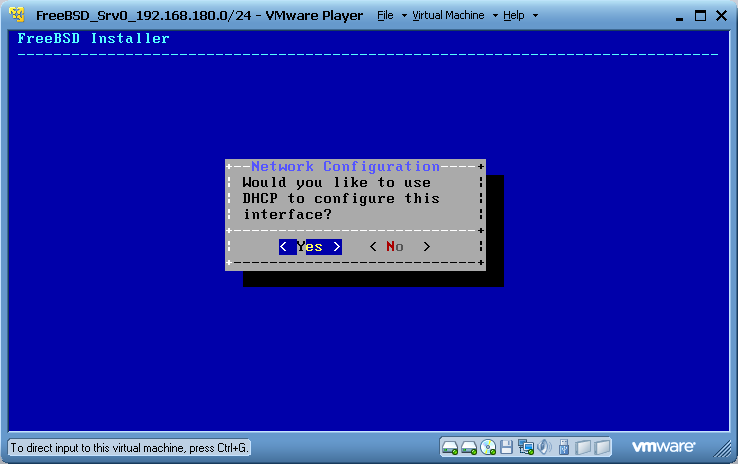


Если же используется беспроводная Wi-Fi сетевая карта, то для подключения необходимо знать SSID и ключ. После выбора беспроводной сетевой карты начнется сканирование на предмет точек доступа (AP). В случае обнаружения, будет выведен список доступных сетей. Если беспроводная сеть защищена паролем, будет необходимо его ввести.

Далее настроим конфигурацию IPv4.



Есть два метода: статический и DHCP. **Отказываемся** от использования DHCP сервера.

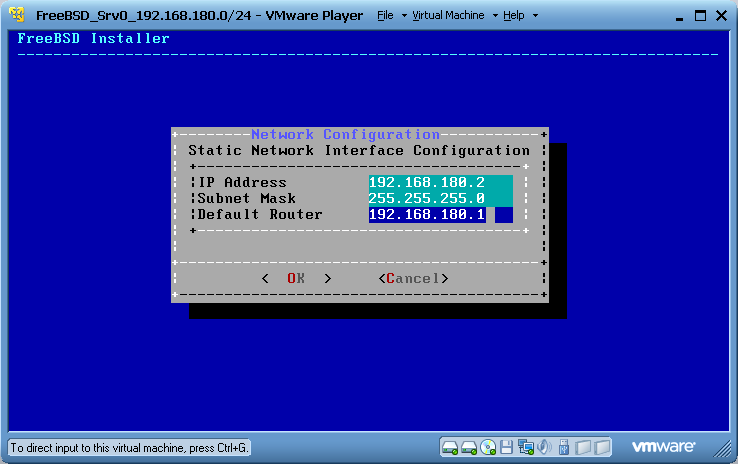


В случае статической конфигурации необходимо ввести параметры настройки сети.

**IP – 192.168.180.X, X – 100+порядковый номер в списке группы.**

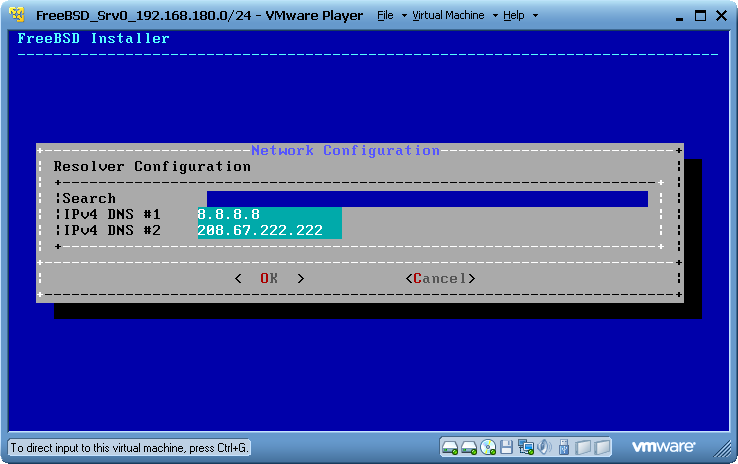
**Маска – 255.255.255.0**

**Default Router (шлюз по умолчанию) – 192.168.180.1**

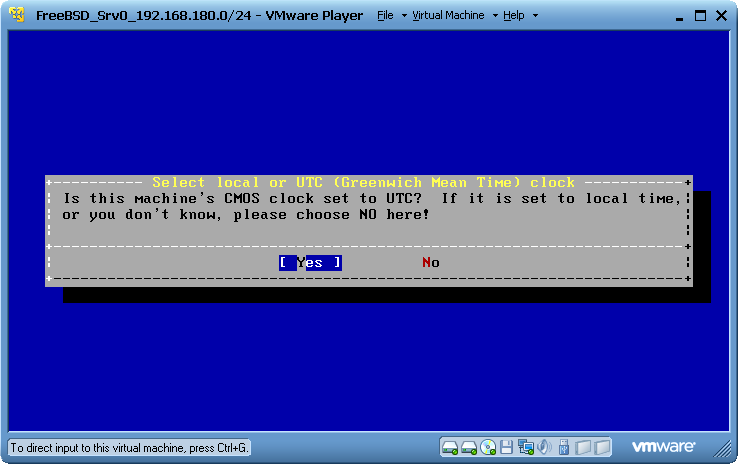


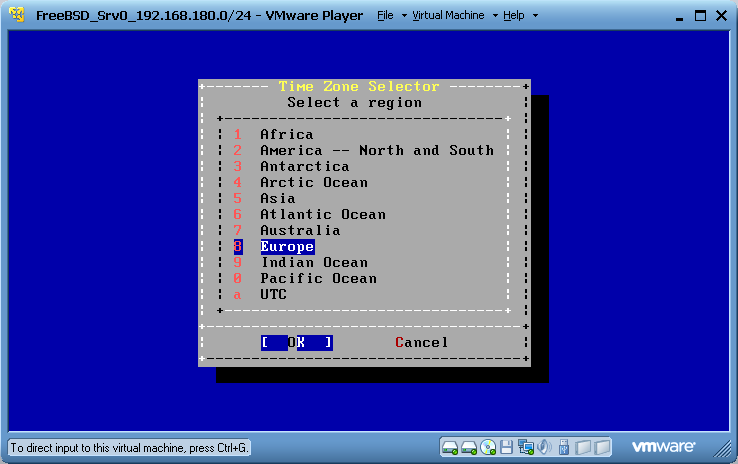
**Настройку IPv6 пропускаем (отказываемся).**

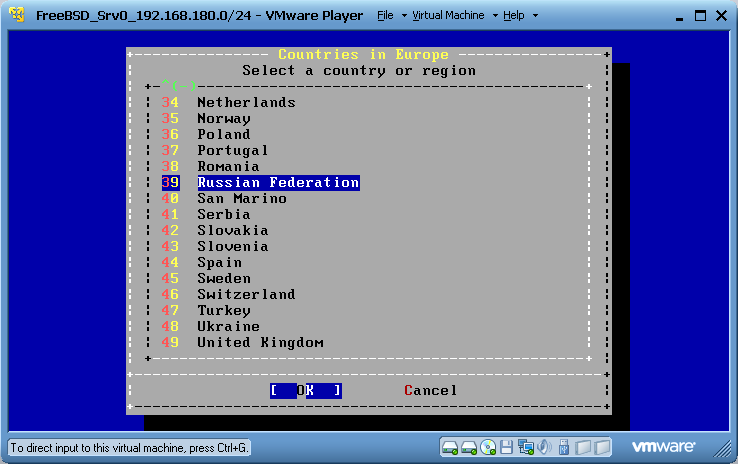
Настраиваем адреса DNS-серверов. Будем использовать публичные DNS сервера.

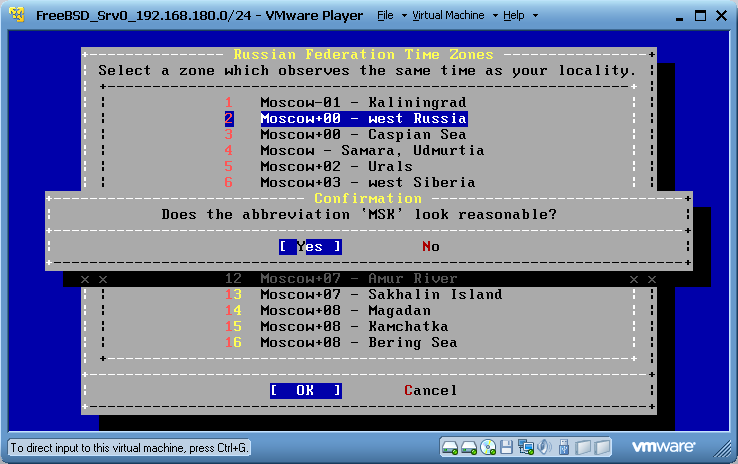


Настраиваем временную зону.



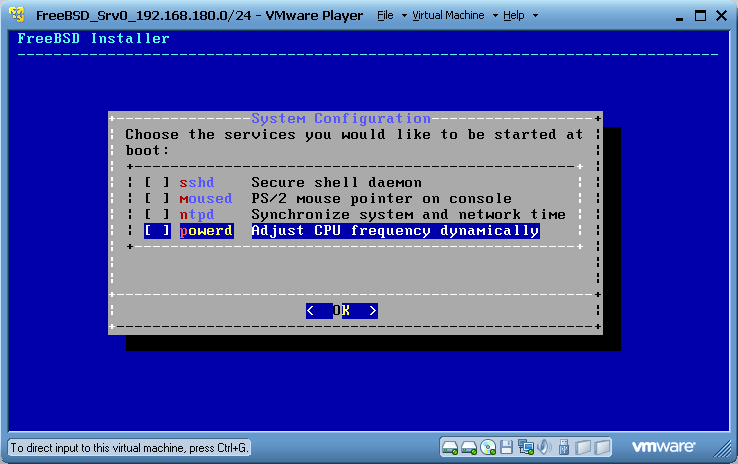




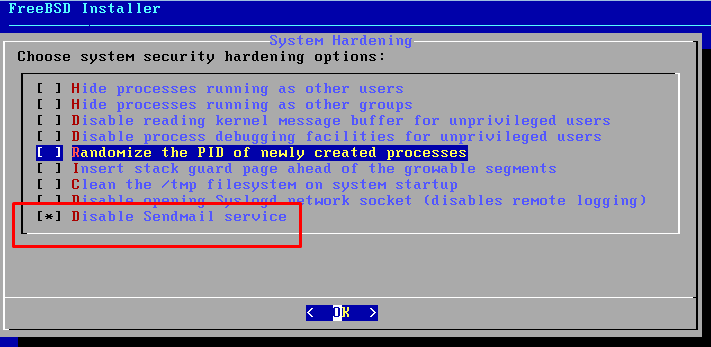


Далее предлагается выбрать сервисы, запускающиеся автоматически при старте системы, **отказываемся от всех, позже настроим самостоятельно**.

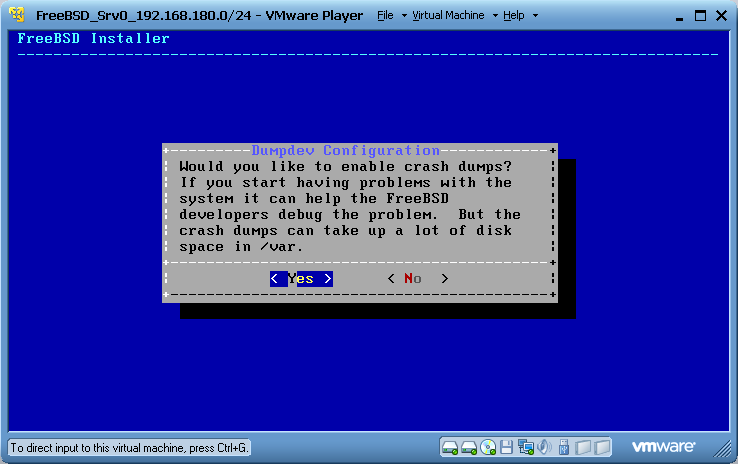
* **sshd** - Secure Shell (SSH).
* **moused** - Поддержка мыши в консоли.
* **ntpd** - Демон Network Time Protocol (NTP) для синхронизации часов через интернет.
* **powerd** - Утилита энергосбережения.



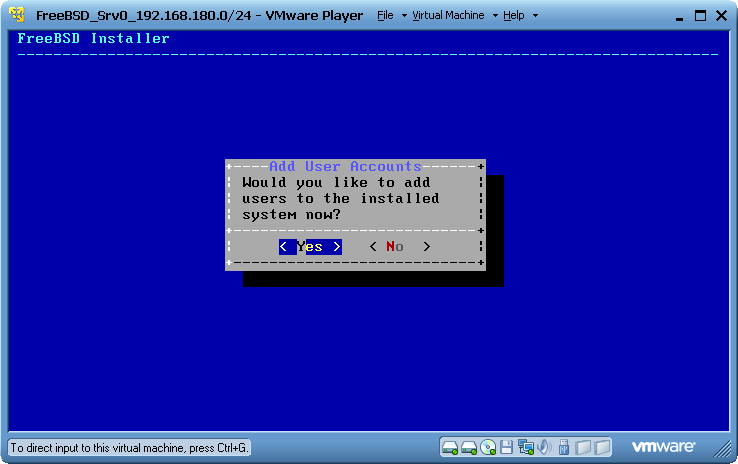
Указываем дополнительные параметры

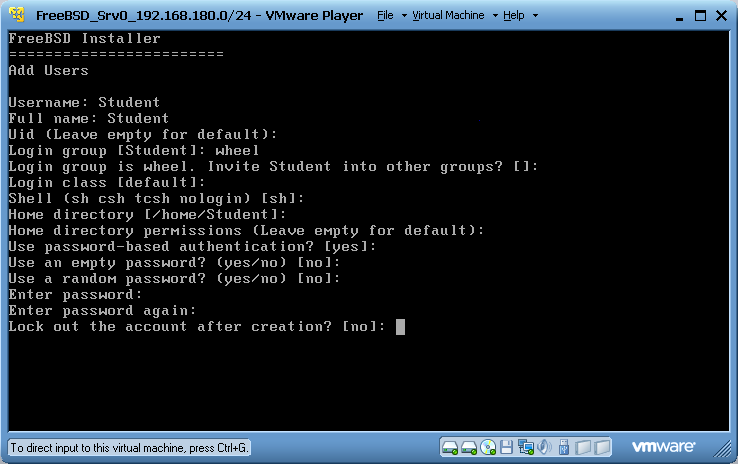


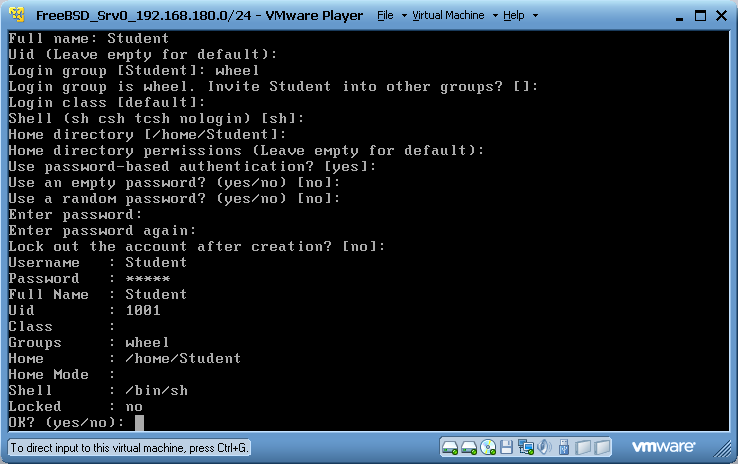
Затем **bsdinstall** поинтересуется - записывать дампы в случае краха системы или нет.



**Добавляем нового пользователя**. Последовательно отвечаем на вопросы (правильные ответы смотрим на картинках).

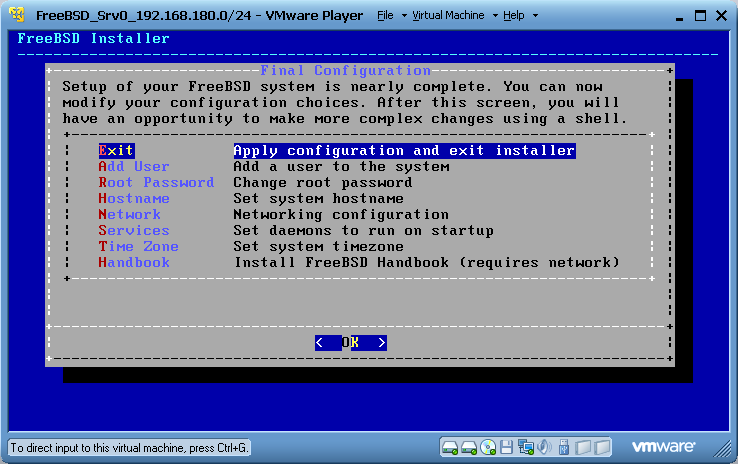


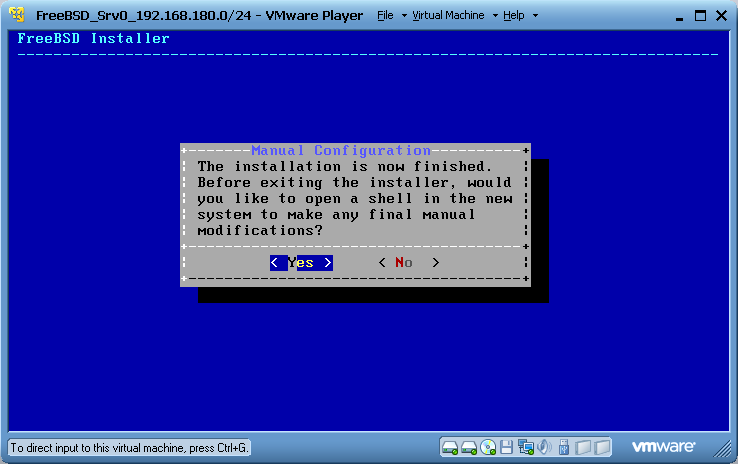




Подтверждаем данные по новому пользователю и отказываемся от заведения еще одного.

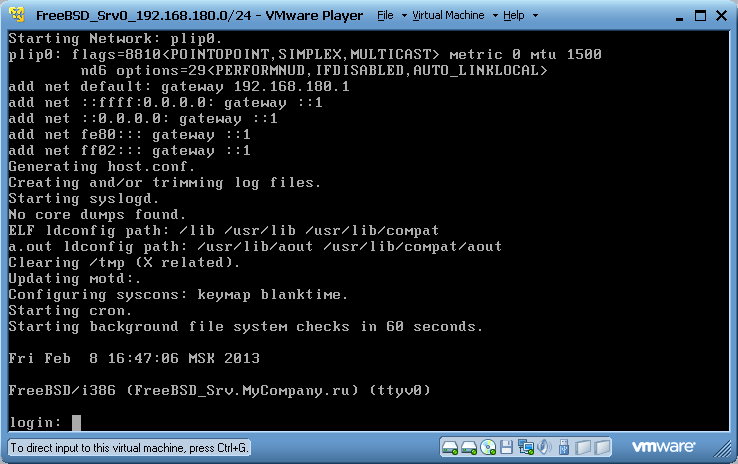
Завершаем установку и выходим из инсталлятора.





Отвечаем «No».

После перезагрузки получаем приглашение ввести логин и пароль пользователя.



**Задание**

Выполнить инсталляцию на виртуальную машину FreeBSD.

Если выполняли все предыдущие пункты, то задание как раз и выполнили.

# Лабораторная работа 3 Работа в ОС FreeBSD. Терминалы, текстовый режим работы. Простые команды. Процессы.

**Цель**

Изучить особенности многопользовательской работы с FreeBSD в текстовом режиме, используя виртуальные терминалы, научиться переключению между терминалами и выполнению команд от имени другого пользователя. Познакомиться и научиться использовать простые команды. Получить представление о процессах, как о способе управления ресурсами. Научиться получать и анализировать информацию о процессах и управлять состоянием выполняющихся процессов.

**Теоретические сведения**

Изначально, терминал - это интерфейс ввода/вывода, состоящий из физических устройства ввода (клавиатура) и вывода (дисплей). Терминал предназначен исключительно для ввода информации и ее отображения на экране. Терминалы бывают физическими (реальными), виртуальными и псевдотерминалами (т.е. программами, которые «притворяются» терминалами). При выполнении лабораторных работ Вы будете использовать виртуальные терминалы.

**Вход в FreeBSD.**

FreeBSD это многопользовательская, многопроцессорная система. Любой многопользовательской системе нужен способ отличать каждого «пользователя» от остальных. В FreeBSD, эта задача решается путем авторизации («входа») пользователя в систему с использованием «персональных» данных. У каждого пользователя есть уникальное имя (''имя пользователя'') и пароль. Перед тем, как разрешить пользователю выполнять какие-либо действия, FreeBSD просит пользователя представиться. В случае правильности (с т.з. системы) введенных данных пользователю предоставляется возможность дальнейшей работы.

**Завершение работы**

FreeBSD требует корректного завершения работы, позволяющего сохранить необходимую системе служебную информацию, завершить в обычном порядке работающие программы, закрыть сеансы и соединения, уведомив об этом всех пользователей.

Для завершения работы во многих UNIX-системах используется утилита **shutdown**. Формат команды: **shutdown** ***action time***.

***action*** - указывает на действие, которое следует совершить,

***-r***(от restart) - перезагрузить компьютер;

***-p***(от power) - выключить питание компьютера;

***time*** - указывает на время совершения события ***action***, например ***now***заставит выполнить утилиту ***shutdown***немедленно.

**Множественные консоли**

При работе в FreeBSD можно использовать несколько виртуальных консолей (терминалов). Вы можете переключаться с одной виртуальной консоли на другую, нажимая соответствующие сочетания клавиш на клавиатуре. У каждой консоли есть свой канал вывода и FreeBSD заботится о том, чтобы правильно перенаправить ввод с клавиатуры и вывод на монитор, как только вы переключитесь с одной консоли на другую.

В конфигурации по умолчанию FreeBSD запускает восемь виртуальных консолей. Тем не менее, это не ограничение оборудования, и вы можете легко настроить систему для загрузки большего или меньшего числа виртуальных консолей. Число и параметры виртуальных консолей задаются в файле ***/etc/ttys***.

Вы можете использовать это файл для настройки виртуальных консолей FreeBSD. Любая не закомментированная строка в этом файле (строка, не начинающаяся с символа #), содержит настройки для одного терминала или виртуальной консоли.

Для переключения между консолями зарезервированы специальные комбинации клавиш. Вы можете использовать сочетания **Alt**-**F1**, **Alt**-**F2**, до **Alt**-**F8**, чтобы переключаться между различными виртуальными консолями в FreeBSD.

При переключении от одной консоли к другой, FreeBSD заботится о сохранении и восстановлении вывода на экран. Программы, которые Вы запускаете на одной виртуальной консоли, не прекращают выполнение, когда консоль становится невидимой, они продолжают выполняться, когда вы переключаетесь на другую виртуальную консоль.

**Процессы. Управление процессами.**

**Процесс** (process) — блок адресного пространства, в котором выполняются одна или более нитей, экземпляр выполняемой программы. Любой процесс может запускать другие процессы. Таким образом, процессы в среде UNIX образуют иерархическую структуру. На вершине этой структуры находится процесс **init**, являющийся предком всех остальных процессов.

**Атрибуты процессов**

С каждым процессом связан набор атрибутов, которые помогают системе контролировать выполнение процессов и распределять между ними ресурсы системы.

**Идентификатор процесса** (process ID) – это целое число однозначно идентифицирующее процесс. Процесс с идентификатором 1 это процесс init. Процесс идентификатором 0 – процесс свопинга (swapping).

**Идентификатор родительского процесса** (parent process ID) указывает на родительский процесс.

**Идентификатор группы процессов** (process group ID). Процессы могут объединяться в группы. Каждая группа обозначается идентификатором группы. Процесс, идентификатор которого совпадает с идентификатором группы, называется лидером группы.

**Идентификатор сеанса** (session ID). Каждая группа процессов принадлежит к сеансу. Сеанс связывает процессы с управляющим терминалом. Когда пользователь входит в систему, все создаваемые им процессы будут принадлежать сеансу, связанному с его текущим терминалом.

Процесс может быть:

* **выполняемым** в текущий момент (R, Runned);
* **находящимся в режиме ожидания** (S, Suspended);
* **прерванным** (T, Terminated), например, при использовании клавиш Ctrl+Z;
* **«зомби»** (Z, Zombied) - завершившимся, но от которого родительский процесс еще не принял сигнала завершения. Спустя некоторое время «зомби» завершаются окончательно и освобождают ресурсы;
* **зависшим**, или в состоянии непрерывного ожидания (uninterruptible sleep). Такой процесс не реагирует на какие-либо сигналы и может быть снят только перезагрузкой системы.

**Идентификаторы пользователя и группы.** С каждым процессом связаны действительные идентификаторы пользователя (real user ID) и группы (real group ID), совпадающие с соответствующими идентификаторами пользователя, запустившего процесс. Кроме того, с процессом связаны эффективные идентификаторы пользователя (effective user ID) и группы, определяющие права процесса в системе. Обычно, действительные и эффективные идентификаторы совпадают.

**Приоритет** (nice) Значение nice показывает готовность процесса уступить свое процессорное время другим процессам. Чем больше значение nice, тем ниже приоритет процесса.

**Даемоны**

Если вы запускаете редактор, им можно легко управлять, открывать в нем файлы и т.д. Вы можете делать это, поскольку редактор предоставляет такие возможности и потому, что редактор присоединен к терминалу. Некоторые программы разработаны без поддержки интерфейса пользователя, поэтому они отсоединяются от терминала при первой возможности. Н-пример, веб-сервер целый день отвечает на запросы из сети, и ему, как правило, не требуется ваше вмешательство. Программы, передающие почту от сервера к серверу – другой пример приложений этого класса.

Мы называем эти программы даемонами (от англ.: daemon). Даемоны (демоны) это персонажи греческой мифологии; хорошие или плохие, они были спутниками человека и, вообще говоря, выполняли полезную работу для людей. Почти как веб- и почтовые серверы выполняют полезную работу сегодня.

Есть соглашение, по которому имя программы, которая обычно запускается как даемон, заканчивается на «d». BIND это Berkeley Internet Name Daemon (выполняемая программа называется named), программа веб-сервера Apache называется httpd, даемон очереди печати это lpd и так далее. Например, главный почтовый даемон для Sendmail называется sendmail.

**Механизмы взаимодействия**

UNIX имеет большое число механизмов межпроцессного взаимодействия. Наиболее популярными средствами являются сигналы, программные каналы (pipes) и именованные каналы (FIFO).

**Сигналы**

Сигналы обеспечивают простой метод прерывания работы процессов. Сигналы используются в основном для обработки исключительных ситуаций. Процесс может определять действия, выполняемые при поступлении сигнала, блокировать сигналы, посылать сигналы другим процессам.

**Основные сигналы:**

SIGCHLD – сигнал о завершении дочернего процесса.

SIGHUP – сигнал освобождения линии. Посылается всем процессам, подключенным к управляющему терминалу при отключении терминала. Многие демоны при получении данного сигнала заново просматривают файлы конфигурации и перезапускаются.

SIGINT – сигнал посылается всем процессам сеанса, связанного с терминалом, при нажатии пользователем клавиши прерывания (CTRL-C).

SIGTERM – сигнал приводит к немедленному прекращению работы получившего сигнал процесса.

SIGKILL – сигнал приводит к немедленному прекращению работы получившего сигнал процесса. В отличие от SIGTERM процесс не может блокировать и перехватывать данный сигнал.

SIGSEGV – сигнал посылается процессу, если тот пытается обратиться к неверному адресу памяти.

SIGSTOP – сигнал приводящий к остановке процесса. Для отправки сигнала SIGSTOP активному процессу текущего терминала можно воспользоваться комбинацией клавиш (CTRL-Z).

SIGCONT – сигнал, возобновляющий работу остановленного процесса.

Для того, чтобы отправить процессу сигнал можно использовать команду **kill**. Для того чтобы процесс мог отправить сигнал другому процессу, необходимо, чтобы эффективные идентификаторы пользователя у посылающего процесса и у процесса получателя совпадали. Процессы с эффективным идентификатором пользователя равным нулю могут посылать сигналы любым процессам.

**Каналы**

Часто возникает ситуация когда два процесса последовательно обрабатывают одни и те же данные. Для обеспечения передачи данных от одного процесса к другому в подобных ситуациях используются программные каналы. Программный канал (pipe) служит для установления связи, соединяющей один процесс с другим. Запись данных в канал и чтение из него осуществляются при помощи системных вызовов write и read, т.е. работа с каналами аналогична работе с файлами. Для создания программного канала используется системный вызов pipe. Вызов возвращает два дескриптора файлов, первый из которых открыт для чтения из канала, а второй для записи в канал. Каналы используются, например, при организации конвейера.

При выполнении команды:

**find /usr/bin -name a\* | sort**

создается канал, команда find выводит в него результаты своей работы, а команда sort считывает из этого канала данные для сортировки.

Главным недостатком программных каналов является то, что они могут использоваться только для связи процессов имеющих общее происхождение (напр., родительский процесс и его потомок). Другой недостаток ограниченное время существования канала (программные каналы уничтожаются после завершения обращающегося к ним процесса). Именованные каналы идентичны программным в отношении записи и чтения данных, но они являются объектами файловой системы. Именованный канал имеет имя, владельца и права доступа. Открытие и закрытие именованного канала осуществляется как открытие и закрытие любого файла, но при чтении и записи он ведет себя аналогично каналу.

**Команды для работы с процессами**

**Top** – консольная команда, которая выводит список работающих в системе процессов и информации о них. По умолчанию она в реальном времени сортирует их по нагрузке на процессор. Чтобы выйти из программы top, необходимо нажать клавишу [q].

Полезные интерактивные команды, которые можно использовать в top:

* **[Пробел]** – Немедленно обновить содержимое экрана.
* **[h]** – Вывести справку о программе.
* **[k]** – Уничтожить процесс. Программа запрашивает у вас код процесса и сигнал, который будет ему послан.
* **[n]** – Изменить число отображаемых процессов. Вам предлагается ввести число.
* **[u]** – Сортировать по имени пользователя.
* **[M]** – Сортировать по объёму используемой памяти.
* **[P]** – Сортировать по загрузке процессора.

**ps [-axewjlu] [-o формат] [-U пользователь] [-p pid]**

Выводит список и статус процессов работающих в системе. Без аргументов выводит список процессов текущего пользователя, подключенных к терминалу, в текущей сессии. Значения параметров следующие:

* **-a** – вывести информацию о процессах всех пользователей.
* **-x** – вывести информацию о процессах, не подключенных к терминалу.
* **-e** – вывести значения переменных окружения процесса.
* **-w** – использовать строки длиной 132 символа. Если указан несколько раз, то строки не обрезаются совсем.
* **-c, -j, -l, -u, -v** – меняют формат вывода информации.
* **-o** – формат вывести информацию в указанном формате.
* **-U** ***пользователь*** – вывести информацию о процессах указанного пользователя.
* **-p** ***pid*** – вывести информацию о процессе с указанным идентификатором.

Значение формата для параметра **-o** является списком из следующих ключевых слов, разделенных запятыми (без пробелов):

* **сommand** командная строка и аргументы.
* **nice** уровень nice (приоритет).
* **pgid** идентификатор группы процессов.
* **pid** идентификатор процесса.
* **ppid** идентификатор родительского процесса.
* **rgid, ruid** реальные идентификаторы группы и пользователя.
* **uid** реальный идентификатор пользователя.
* **tty** управляющий терминал

**kill [-s signal| -signal] pid**

Посылает сигнал указанному процессу. Если значение сигнала опущено, предполагается SIGTERM. signal – символическое имя сигнала без префикса SIG, либо номер сигнала.

Пример: **kill -HUP 172** – послать сигнал SIGHUP процессу с идентификатором 172.

Выполнение **kill -s KILL 1** – быстрый способ перегрузить систему, т.к. главный процесс (**init**) имеет идентификатор **1**.

**nice [-nice] команда [аргументы]**

Выполняет команду с меньшим приоритетом. Если nice не задан, то предполагается значение nice может быть от -20 (наивысший приоритет) до 20 (наименьший приоритет). Отрицательные числа задаются как –nice. Увеличение приоритета может выполнить только суперпользователь.

Пример: **nice -10 john users** – запустить программу john с пониженным приоритетом.

**Перенаправление стандартного вывода**

Для обозначения перенаправления используются символы **">"**, **"<"** и **">>"**. Чаще всего используется перенаправление вывода команды в файл. Вот соответствующий пример:

**ls -l > /home/Student/dir.txt**

По этой команде в файле **/home/ Student/dir.txt** будет сохранен перечень файлов и подкаталогов того каталога, который был текущим на момент выполнения команды **ls**, при этом если указанного файла не существовало, то он будет создан; если он существовал, то будет перезаписан; чтобы вывод команды был дописан в конец существующего файла, то надо вместо символа **>** использовать **>>**.

Новый текстовый файл можно создать, например, с помощью перенаправления стандартного ввода с клавиатуры в файл:

**cat > f1.txt**

**Hello World!**  // для завершения ввода необходимо нажать Ctrl+C

или перенаправить вывод команды в файл

**echo “Privet Student!” > f2.txt**

Можно и посмотреть созданный файл с помощью команды: **cat f1.txt**

Команды **more** и **less**

**more** – это основная команда постраничной или построчной прокрутки большого текста на экране и поиска в этом тексте.

После нажатия клавиши Enter на экране появится первая страница текста, в последней строке которой будет одно слово -More- (еще). Нажатие клавиши пробела прокручивает текст на одну страницу, нажатие Enter - на одну строку. В конце текста на экране появится подсказка командной строки.

Команде **more** можно передать результаты выполнения другой команды, воспользовавшись методом конвейеризации.

**ls -l /usr/ports | more**

**less** – усовершенствованный вариант команды **more**. В дополнение к функциям, описанным выше (постраничная или построчная прокрутка текста от начала до конца и поиск), команда less позволяет выполнять следующие операции:

* переход на указанную строку;
* переход в начало или конец текста;
* прокрутка текста от конца к началу;
* поиск в обратном направлении.

Чтобы перейти на определенную строку текста, введите ее номер с буквой **g** в конце. Если опустить номер строки, будет выполнен переход к первой строке. Вместо строчной буквы **g** можно указать заглавную (**G**), но тогда при отсутствии номера строки будет выполняться переход не к первой, а к последней строке.

Для построчной прокрутки текста вверх или вниз достаточно нажать соответствующую клавишу со стрелкой. Постраничная прокрутка вверх выполняется нажатием клавиш **Ctrl+B**. Для выхода нажмите **q**.

**Выполнение**

Сразу после загрузки FreeBSD и завершения работы стартовых скриптов, система предлагает ввести имя пользователя:

**login:**

В этом примере, будем использовать предопределенное имя «главного» пользователя - root. Вводим *root* в приглашении и нажимаем **Enter**. Далее должно появиться приглашение ввести «пароль»:

**login: root**

**Password:**

Введите соответствующий имени root пароль и нажмите **Enter**. Пароль *не виден*! Не беспокойтесь об этом. Это сделано по соображениям безопасности. Если вы ввели пароль правильно, то сразу же войдете в FreeBSD и можете начать выполнять команды. Вы увидите сообщение дня (MOTD, или message of the day, которое хранится в текстовом файле /etc/motd) за которым последует командная строка (с символом *#*, *$*, или *%*). Это означает, что вы успешно вошли в FreeBSD.

Информацию о командах вы можете получить через встроенную справочную систему формата **man** (от manuals) или **info**. Для получения справки достаточно ввести **man** (или **info**) с именем нужной команды в качестве параметра. Получите справку по команде **pwd**.

Изучите справку по простым командам:

* **pwd** – возвращает имя текущей директории
* **tty** – возвращает имя текущего терминала
* **who** – возвращает список пользователей вошедших в систему
* **w** – немного более подробная информация о работающих пользователях
* **uname** – возвращает информацию о системе
* **uptime** – возвращает время работы системы
* **date** – сообщает или устанавливает дату и время
* **exit** – завершение сеанса
* **ls [-F|-l|-a]** – возвращает содержание директории; **-F** выдает список файлов в текущей директории со значком «\*» после исполняемых файлов, «/» – после директорий и «@»– после символических ссылок. Ключ **-a** – все файлы, в том числе скрытые; **-l** – в длинном формате, с указанием даты и прочего.
* **cd [..|~]** ***имя\_директории*** – позволяет изменить текущую директорию
* **cat *filename*** – выдаёт содержимое ***filename*** на экран, если он слишком длинный и вы можете увидеть только его конец, нажмите ScrollLock и используйте клавишу стрелка вверх для движения назад; вы можете также использовать ScrollLock и со страницами справки. Нажмите ScrollLock снова для прекращения прокрутки

Используя простые команды, выполните следующее:

1. выясните, какая директория является текущей
2. выясните, в каком терминале выполняется текущий сеанс
3. откройте новый сеанс в **ttyv3** для пользователя **Student**
4. повторите пункты 1-2
5. перейдите в **ttyv0**, выясните, какие пользователи и в каких терминалах работают
6. при помощи команд **tty**, **w**, **uname**, **uptime** выведите в файл **info.txt** имя текущего терминала, информацию о пользователях, работающих в системе, название и версию операционной системы, время работы системы
7. завершите сеанс в **ttyv3**

Используя теоретические сведения и справочную систему FreeBSD, выполните следующее:

1. войдите в систему с учетной записью **Student** (например, в терминале **ttyv2**)
2. получить справку о команде **ps**
3. командой **ps** выведите краткую информацию о выполняющихся процессах в текущем терминале и определите PID текущей оболочки
4. получите подробную информацию о загруженных процессах, какой из них использует максимальный объем памяти, а какой - максимально загружает процессор
5. по информации, полученной в пункте 4, выясните, какой PID имеет процесс **init** и от чьего имени он запущен
6. изучите ключи **-c, -v, -j, -u, -l**, которые изменяют формат вывода команды; выполните команду **ps** с каждым из этих ключей, результаты сохраните в соответствующих файлах
7. определите, какие процессы порождены процессом **init**
8. работая под учетной записью **root**, выполните команду **top**
9. переключитесь в терминал, где работаете от имени **Student**
10. найдите в списке процессов идентификатор процесса **root** для команды **top**
11. используя команду **kill**, попытайтесь «убить» этот процесс, объясните результат
12. запустите команду **top** от имени **Student** и попробуйте «убить» это процесс, работая от имени **root**
13. работая под учетной записью **Student**, выведите в файл отчета (**ps.txt**) следующую информацию о запущенных **Вами** процессах: идентификатор процесса, терминал, идентификатор пользователя, приоритет, команда
14. работая под учетной записью **Student**, выведите в файл отчета (**ps1.txt**) информацию о процессах, запущенных пользователем **root**

Выключите виртуальную машину, работая от имени Student. Если не удалось, попробуйте от имени **root**. Для этого используйте команду:

**shutdown –p now**

# Лабораторная работа 4 Работа в ОС FreeBSD. Файловая система. Управление каталогами.

**Цель**

Изучить особенности файловой системы FreeBSD, освоить работу с основными командами манипулирования файлами и каталогами, научиться монтировать файловые системы.

**Теоретические сведения**

Файловая система FreeBSD является ключевым моментом в понимании устройства всей системы. Самым важным понятием является, корневой каталог, обозначаемый символом **«/»**. Корневой каталог *монтиру­ется* самым первым на этапе загрузки и содержит все необходимое, чтобы подготовить систему к загрузке в многопользовательский режим. Корне­вой каталог также содержит точки монтирования всех других файловых систем.

**Точкой монтирования** – каталог, который будет соответствовать корню примонтированой файловой системы. Стандартные точки монтиро­вания включают /usr, /var, /tmp, /mnt и /cdrom. Эти каталоги обыч­но перечислены в файле **/etc/fstab**, в котором указаны файловые сис­темы и их точки монтирования.

|  |  |
| --- | --- |
| **Каталог** | **Описание** |
| / | Корневой каталог файловой системы. |
| /bin/ | Основные утилиты, необходимые для работы как в одно­пользовательском, так и в многопользовательском режимах. |
| /boot/ | Программы и конфигурационные файлы, необходимые для нормальной загрузки операционной системы. |
| /boot/defaults/ | Конфигурационные файлы с настройками по умолчанию, ис­пользуемые в процессе загрузки операционной системы. |
| /dev/ | Файлы устройств. |
| /etc/ | Основные конфигурационные файлы системы и скрипты. |
| /etc/defaults/ | Основные конфигурационные файлы системы с настройками по умолчанию. |
| /etc/mail/ | Конфигурационные файлы для систем обработки почты. |
| /etc/namedb/ | Конфигурационные файлы **ДЛЯ** службы DNS named. |
| /etc/periodic/ | Файлы сценариев, выполняемые ежедневно, еженедельно и ежемесячно. |
| /etc/ppp/ | Конфигурационные файлы для утилиты ppp. |
| /mnt/ | Пустой каталог, используемый как временная точка монти­рования. |
| /proc/ | Виртуальная файловая система, отображающая текущие про­цессы. |
| /root/ | Домашний каталог пользователя root. |
| /sbin/ | Системные утилиты и утилиты администрирования, необхо­димые для работы, как в однопользовательском, так и в мно­гопользовательском режимах. |
| /stand/ | Программы, необходимые для работы в автономном режиме (например, при установке системы). |
| /tmp/ | Временный каталог. |
| /usr/ | Большинство пользовательских утилит и приложений. |
| /usr/bin/ | Пользовательские утилиты и приложения общего назначе­ния. |
| /usr/include/ | Стандартные заголовочные файлы для языка С. |
| /usr/lib/ | Файлы стандартных библиотек. |
| /usr/libdata/ | Файлы данных для различных утилит. |
| /usr/libexec/ | Системные даемоны и утилиты (выполняемые другими про­граммами). |
| /usr/local/ | Локальные пользовательские приложения, библиотеки, и т.д. |
| /usr/obj/ | Архитектурно-зависимые файлы и каталоги, образующиеся в процессе сборки системы из исходных текстов в /usr/src. |
| /usr/sbin/ | Системные утилиты и утилиты администрирования (испол­няемые пользователем). |
| /usr/share/ | Архитектурно-независимые файлы. |
| /usr/src/ | Исходные тексты BSD и/или программ. |
| /usr/X11R6/ | Утилиты, приложения и библиотеки X11R6 (X Window System). |

**Организация дисков**

Наименьшая единица, которую FreeBSD использует для обращения к фай­лам, это имя файла. Имена файлов чувствительны к регистру, поэтому readme.txt и README.TXT — два разных файла. FreeBSD не использу­ет расширение файла (.txt) для определения программа это, документ или другой тип данных.

Файлы хранятся в каталогах. Каталоги могут не содержать файлов, или мо­гут содержать много сотен файлов. Каталоги также могут содержать дру­гие каталоги, что позволяет создавать иерархию каталогов один в другом. Это упрощает организацию данных.

Обращение к файлам происходит путем задания имени файла или каталога, дополняемого прямым слэшем /, за которым может следовать имя другого каталога. Если есть каталог *Каталог\_Первый*, содержащий каталог *Каталог\_Второй*, который со­держит файл *readme.txt*, полное имя, или *путь* к файлу будет *Каталог\_Первый*/*Каталог\_Второй*/*readme.txt*.

Каталоги хранятся в файловых системах. Каждая файловая система содер­жит один каталог на верхнем уровне, называемый *корневым каталогом* этой файловой системы. Этот корневой каталог может содержать другие каталоги.

FreeBSD не использует букв дисков, или других имен дисков в пути. Вместо этого, одна файловая система назначается *корневой файловой сис­темой.* Обращение к корневому каталогу корневой файловой системы происходит через /. Любая другая файловая система *монтируется* к кор­невой файловой системе. Неважно как много дисков есть в вашей системе FreeBSD, каждый каталог будет выглядеть как расположенный на том же диске.

**Преимущества нескольких файловых систем**

Различные файловые системы могут иметь различные *опции монтирова­ния.* Например, в целях безопасности корневая файловая система может быть смонтирована только для чтения, что делает невозможным случайное удаление или редактирование критически важного файла. Отделение фай­ловых систем, используемых пользователями для записи, таких как **/home**, от других файловых систем позволяет также монтировать их с параметром *nosuid;* этот параметр отменяет действие битов *suid/guid* на исполняемых файлах, в этой файловой системе, что потенциально повышает безопас­ность.

FreeBSD автоматически оптимизирует расположение файлов на файловой системе в зависимости от того, как файловая система используется. Фай­ловая система, содержащая множество мелких часто записываемых файлов, будет иметь оптимизацию, отличную от таковой для файловой системы, содержащей несколько больших файлов.

**Монтирование и размонтирование файловых систем**

Файловая система лучше всего представима в виде дерева, с корнем в /. Каталоги, /dev, /usr и прочие - это ветви дерева, которые, в свою оче­редь, являются корнями для поддеревьев, также имеющих ветви (/usr/local), и т.д.

Хорошей практикой является разнесение некоторых особо важных катало­гов на разные файловые системы. Например, /var, содержит log/, spool/, а также всевозможные временные файлы и нередко увеличивает­ся в размерах настолько, что занимает все свободное место на диске. По­этому лучше смонтировать /var отдельно, чтобы избежать переполнения корневой директории /.

**Команда mount**

Команда mount(8) используется для монтирования файловых систем:

**mount устройство точка\_монтирования,**

где **устройство** - имя монтируемого устройства;

**точка\_монтирования** - директория, в которую будет вмонтирована файловая система устройства.

Опции, которые может принимать команда mount:

* **-r** – монтировать файловую систему в режиме "только для чтения"
* **-w** – монтировать файловую систему в режиме "чтение-запись"
* **-t *fstype*** – монтировать файловую систему как систему указанного типа, по умолчанию, тип файловой системы – «ufs»

Для монтирования специальных файловых систем, например файловых систем Microsoft, существуют модификации команды mount, например:

**mount\_msdosfs** – для монтирования систем FAT16 и FAT32;

**mount\_ntfs** – для монтирования систем NTFS;

Для некоторых общеизвестных устройств команда mount принимает упрощенный вид, например, mount /cdrom, mount /mnt/floppy. Для других устройств, их имена необходимо указывать в явном виде, на­пример: /dev/ad0s**N** - для жестких дисков, /dev/da0s**N** - для USB Flash-карт, /dev/fd**N** - для приводов гибких дисков и т.д. Здесь **N** - порядковый номер устройства текущего типа. Посмотреть наличие устройств и их точные имена можно в директории /dev.

**Команда umount**

Команда umount(8) принимает в качестве параметра **точку монтирования** какой-либо файловой системы либо **имя устройства** либо опцию *-а* или *-A.* Кроме того, вы можете дополнительно указать опцию *-f* для форсирован­ного размонтирования файловой системы, и *-v* для получения более под­робной информации. Имейте ввиду, что это в общем случае опасно и по­тому не рекомендуется, так как тем самым вы можете нарушить работу компьютера или повредить данные на файловой системе. Опции *-а* и *-A* используются для размонтирования всех файловых систем (разве что вы укажете опцию *-t).* Разница состоит в том, что *-A* не пытается размонтировать корневую файловую систему.

**Команда df**

Позволяет посмотреть свободное место на дисках.

Утилита df выводит данные о размере свободного дискового пространства указанной файловой системы или файловой системы, к которой относится указанный файл. Выводимые значения соответствуют количеству 512-байтных блоков. сли не заданы ни файл, ни файловая система, утилита выводит статистику по всем cмонтированным файловым системам (зависит от опции -t, смотрите её описание ниже).

Имеются следующие опции:

* -a – Показывать все точки монтирования, включая те, что были смонтированы с флагом MNT\_IGNORE.
* -b – Использовать для вывода блоки по 512 байт.
* -c – Показывать общую сумму.
* -g – Использовать для вывода блоки по 1073741824 байт (1 Гбайт).
* -H – "Удобочитаемый" вывод. Использовать суффиксы B (байт), K (килобайт), M (мегабайт), G (гигабайт), T (терабайт) и P (петабайт) для уменьшения количества цифр в результате до 4-х или менее, используя для этого соответствующую степень числа 10.
* -h – "Удобочитаемый" вывод. Использовать суффиксы B (байт), K (килобайт), M (мегабайт), G (гигабайт), T (терабайт) и P (петабайт) для уменьшения количества цифр в результате до 4-х или менее, используя для этого соответствующую степень числа 2.
* -i – Дополнить отчёт количеством свободных индексных дескрипторов (inodes).
* -k – Использовать для вывода блоки по 1024 байт (1 Кбайт).
* -l – Отображать данные только о локально-смонтированных файловых системах.
* -m – Использовать для вывода блоки по 1048576 байт (1 Мбайт).
* -n – Показывать ранее полученную статистику по файловым системам.
* -P – Указывает на то, что вывод должен соответствовать стандарту POSIX. Выводимые значения будут в 512-байтных блоках.
* -t – Отображать данные только о файловых системах заданного типа. Можно задать несколько типов, разделив их список запятыми.

**Команды управления файловой системой**

Для управления файловой системой имеются различные команды, реализующие операции по созданию, чтению, копированию, переименованию/перемещению, изменению и удалению файлов и каталогов. Как правило, это специализированные команды, хорошо выполняющие свою задачу, однако некоторые функции могут частично дублироваться другими командами, что только добавляет гибкости управлению файлами.

Основными командами для выполнения файловых операций являются: **pwd, ls, cp, mv, rm, cd, rmdir, mkdir, ln**.

**Команда cp**

Позволяет копировать файлы.

Синтаксис:

cp [-R [-H | -L | -P]] [-f | -i | -n] [-lpv] исходный\_файл целевой\_файл

cp [-R [-H | -L | -P]] [-f | -i | -n] [-lpv] исходный\_файл целевой\_каталог

В первой форме утилита **cp** копирует содержимое файла исходный\_файл в целевой\_файл. Во второй форме содержимое каждого указанного файла копируется в целевой\_каталог. Имена самих файлов при этом не изменяются. Если **cp** обнаруживает попытку копирования файла самого в себя, копирование не производится.

Имеются следующие опции:

* -H –Если указана опция -R, следовать символическим ссылкам, указанным в командной строке, но не следовать символическим ссылкам, встречающимся в процессе обхода дерева каталогов.
* -L – Если указана опция -R, следовать всем символическим ссылкам.
* -P – Если указана опция -R, не следовать никаким символическим ссылкам. Это режим работы по умолчанию.
* -R – Если исходный\_файл указывает на каталог, **cp** копирует этот каталог вместе со всем деревом файловой иерархии, которое он содержит. Если исходный\_файл оканчивается на /, копируется не сам каталог, а только его содержимое. Эта опция также указывает на необходимость копирования символических ссылок, а не объектов, на которые они указывают, и заставляет **cp** создавать специальные файлы вместо того, чтобы копировать их, как обычные. Создаваемые каталоги имеют те же права доступа, что и исходные каталоги, вне зависимости от действующего для процесса значения umask. Обратите внимание, что **cp** копирует жёсткие ссылки как отдельные файлы.
* -f – Для каждого уже существующего целевого имени пути, удалять его и создавать новый файл, не требуя подтверждения операции, вне зависимости от его прав доступа.
* -i – Заставляет **cp** выдавать запрос через стандартный файл ошибок перед копированием файла, которое бы вызвало перезапись существующего файла. Если ответ, полученный со стандартного ввода, начинается символом `y' или `Y', будет предпринята попытка копирования.
* -l – Создавать жёсткие ссылки на обычные файлы в иерархии вместо того, чтобы их копировать.
* -n – Не перезаписывать существующие файлы.
* -p – Заставляет **cp** сохранять следующие атрибуты каждого исходного файла в создаваемой копии: время модификации, время доступа, флаги файла, режим доступа, идентификаторы пользователя и группы, насколько это позволено правами доступа. Если значения идентификаторов пользователя и группы не могут быть сохранены, не выводится никакого сообщения об ошибке, и возвращаемое программой значение статуса завершения не изменяется.
* -v – Выводить больше информации, показывая файлы по мере их копирования. Для каждого уже существующего целевого файла, его содержимое перезаписывается, если позволяют права. Его режим доступа и идентификаторы пользователя и группы не изменяются, если не задана опция -p.

**Команда mkdir**

Позволяет создать каталоги

Синтаксис

mkdir [-pv] [-m режим] имя\_каталога

Имеются следующие опции:

* -m – Установить права доступа конечного создаваемого каталога в соответствии с указанным режимом. Аргумент режим может быть задан в любом из форматов, определённых для команды chmod(1). Если используется символьный режим, то символы `+' и `-' интерпретируются относительно начального режима ``a=rwx''.
* -p – Создавать промежуточные каталоги по мере необходимости. Если эта опция не указана, то префикс полного пути каждого из операндов должен уже существовать. С другой стороны, если эта опция указана, то существование каталога, заданного операндом, не является ошибкой.
* -v – Выводить подробную информацию при создании каталогов, показывая их по мере создания. Пользователь должен иметь право на запись в родительский каталог.

**Команда mv**

Позволяет переместить файлы

Синтаксис

mv [-f | -i | -n] [-v] источник цель

mv [-f | -i | -n] [-v] источник каталог

В первом варианте вызова утилита mv переименовывает файл, заданный аргументом источник, в целевой путь, заданный аргументом цель. Такая форма подразумевается, когда последний операнд не является именем уже существующего каталога.

Во втором варианте mv перемещает каждый файл источник в целевой файл в существующем каталоге, заданным операндом каталог. Целевой путь для каждого операнда получается конкатенацией последнего операнда, косой черты, и заключительной части имени пути заданного файла.

Имеются следующие опции:

* -f – Не запрашивать подтверждение перед перезаписью целевого пути.
* -i – Заставляет mv выдавать запрос через стандартный файл ошибок перед перемещением файла, которое бы вызвало перезапись существующего файла. Если ответ со стандартного ввода начинается символом `y' или `Y', то будет сделана попытка выполнить перемещение.
* -n – Не перезаписывать существующие файлы.
* -v – Выводить больше информации, показывая файлы по мере их перемещения.

Указание каталога в качестве операнда источник является ошибкой, если цель существует, и не является каталогом.

Если права доступа к целевому пути не позволяют выполнить запись, то mv запросит подтверждение у пользователя, как если бы была указана опция -i.

Поскольку системный вызов rename(2) работает только в пределах одной файловой системы, mv использует cp(1) и rm(1) для выполнения перемещения.

Эффект аналогичен выполнению последовательности команд:

rm -f целевой\_путь && \

cp -pRP файл\_источник цель && \

rm -rf файл\_источник

**Команда rm**

rm, unlink – удалить элементы каталога

Синтаксис:

rm [-f | -i] [-dIPRrvW] файл ...

unlink файл

Утилита **rm** пытается удалить файлы, указанные в командной строке и не являющиеся каталогами. Если права доступа к файлу не разрешают запись и устройством стандартного ввода является терминал, то пользователю выводится запрос (через стандартный файл ошибок) на подтверждение действия.

Имеются следующие опции:

* -d – Пытаться удалять каталоги так же, как и файлы других типов.
* -f – Пытаться удалять файлы без запроса подтверждения, вне зависимости от установленных прав доступа к файлу. Если файл не существует, не выводить никакой диагностической информации и не менять статус завершения для информирования об ошибке.
* -i – Запрашивать подтверждение перед удалением каждого файла, несмотря на установленные права доступа к файлу, и вне зависимости от того, является ли устройство стандартного ввода терминалом или нет.
* -I – Запрашивать подтверждение только если удалению подлежит более трёх файлов, или перед рекурсивным удалением каталога. Установка этой опции позволяет существенно снизить количество запросов (по сравнению с опцией -i), и в то же время обеспечивает практически такой же уровень защиты от ошибок.
* -P – Перезаписывать обычные файлы перед удалением. Файлы перезаписываются трижды, сначала значением 0xff, затем 0x00, затем ещё раз 0xff, и только потом удаляются. Файлы с несколькими жёсткими связями не будут ни перезаписаны, ни удалены (если только не указана опция -f), а вместо этого будет выводиться предупреждение. Указание этого флага для файла, имеющего доступ только на чтение, приведёт к выводу сообщения об ошибке и завершению работы rm. Файл при этом не будет перезаписан или удалён.
* -R – Пытаться удалить дерево каталогов, корень которого указан аргументом файл.
* -r – То же самое, что и -R.
* -v – Выводить больше информации при удалении файлов, показывая их по мере их удаления.
* -W – Пытаться восстановить перечисленные файлы.

Утилита **rm** удаляет символические ссылки, а не сами файлы, на которые эти ссылки указывают. Попытка удалить файлы **/, .** или **..** вызовет ошибку.

**Команда rmdir**

Позволяет удалить каталоги

Синтаксис

rmdir [-pv] каталог

Утилита **rmdir** удаляет каталоги, заданные аргументами каталог, в том случае, если они пусты.

Аргументы обрабатываются в указанном порядке, поэтому для удаления и родительского каталога, и его подкаталога вначале должен быть указан подкаталог, чтобы родительский каталог был уже пуст на момент, когда **rmdir** попытается его удалить.

Имеются следующие опции:

* -p – Каждый каталог рассматривается как путь, все компоненты которого будут удалены, если они пусты, начиная с последнего компонента.
* -v – Выводить подробную информацию, показывая каталоги по мере их удаления.

**Команда ln**

Позволяет создать ссылки

Синтаксис:

ln [-Ffhinsv] исходный\_файл [целевой\_файл]

ln [-Ffhinsv] исходный\_файл ... целевой\_каталог

link исходный\_файл целевой\_файл

Утилита **ln** создаёт в каталоге новую запись (ссылку на файл), все характеристики которой совпадают с исходным файлом. Это позволяет иметь одновременно несколько копий файла в разных местах, не тратя при этом место на диске для копий; вместо этого ссылка «указывает» на оригинал. Существует два типа ссылок: **жёсткие** и **символические**. Каким образом ссылка «указывает» на файл, зависит от типа ссылки.

Имеются следующие опции:

* -f – Если целевой файл уже существует, удалить его, чтобы можно было создать ссылку.
* -F – Если целевой файл уже существует и является каталогом, удалить его, чтобы можно было создать ссылку. Опцию -F следует использовать с опциями -f или -i, и если ни одна из них не указана, то подразумевается опция -f. Опция -F ничего не делает, если не указана опция -s.
* -h – Если целевой\_файл или целевой\_каталог является символической ссылкой, не следовать ей. Эта опция особенно полезна в сочетании с опцией -f для замены символической ссылки, которая может указывать на каталог.
* -i – Заставляет ln выдавать запрос через стандартный файл ошибок в случае, если целевой файл существует. Если ответ, полученный со стандартного ввода, начинается символом `y' или `Y', удалить целевой файл, чтобы можно было создать ссылку. В противном случае, не пытаться создавать ссылку.
* -n – То же, что и -h, для совместимости с другими реализациями ln.
* -s – Создать символическую ссылку.
* -v – Выводить больше информации, показывая обрабатываемые утилитой файлы.

По умолчанию **ln** создаёт жёсткие ссылки. Жёсткая ссылка на файл ничем не отличается от исходного файла; при этом любые изменения в файле совершенно не зависят от имени, по которому к нему обратились. Жёсткие ссылки не могут ссылаться на каталоги, и не могут находиться на другой файловой системе.

Символическая ссылка содержит имя файла, на который она ссылается. При выполнении операции open(2) над символической ссылкой используется оригинальный файл. Вызов stat(2), выполненный над символической ссылкой, также вернёт исходный файл. Для получения информации о ссылке следует использовать lstat(2). Для чтения содержимого символической ссылки можно воспользоваться вызовом readlink(2). Символические ссылки могут находиться на другой файловой системе и могут указывать на каталоги.

Важно понимать разницу между «жесткими» и «мягкими» ссылками.

|  |  |
| --- | --- |
| **«Жесткие» ссылки** | **«Мягкие» ссылки** |
| Возможны только в рамках одной физической файловой системы (в рамках одного дискового раздела) | Возможны на любые объекты в рамках файловой системы (даже между дисковыми разделами) |
| Указывают на саму область данных в файловой системе | Указывает не на сами данные, а на символьное имя файла внутри файловой системы, т.е. на одну существующую «жесткую» ссылку или на другую «мягкую» ссылку |
| Все «жесткие» ссылки на один файл первичны и равнозначны | Является вторичной ссылкой по отношению к объекту файловой системы, на символьное имя которого она указывает |
| При удалении одной из нескольких «жестких» ссылок на один и тот же файл сам файл не удаляется, все остальные «жесткие» ссылки продолжают существовать и обеспечивать к нему доступ. Сам файл перестает существовать только после удаления последней «жесткой» ссылки на него.  При первичном создании файла счетчик его «жестких» ссылок устанавливается в «1». При добавлении «жесткой» ссылки на тот же файл счетчик увеличивается, при удалении счетчик уменьшается. При обнулении счетчика данные помечаются удаленными. | При удалении «мягкой» ссылки сам файл продолжает существовать.  При удалении «жесткой» ссылки, на символьное имя которой указывает «мягкая» ссылка, эта «мягкая» ссылка продолжает существовать, но становиться «не рабочей» и не обеспечивает доступ к данным (даже если сами данные не удалены и доступны по другим «жестким» ссылкам). |

Рассмотрим пример. Создадим файл file\_a с содержимым «aaa»:

**echo "aaa" > file\_a**

Создадим на него 2 ссылки: символическую и жёсткую:

**ln -s file\_a link1**

**ln file\_a link2**

Попробуем достучаться до файла по обеим ссылкам:

**cat link1**

получим - **aaa**

**cat link2**

получим - **aaa**

Теперь удалим оригинальный файл и попробуем достучаться ещё раз. Первая ссылка окажется «битой», а по второй ссылке по-прежнему находится файл с содержимым «aaa»:

**rm file\_a**

**ls**

получаем - **link1 link2**

**cat link1**

**cat: link1**

получаем - **cat: link1: No such file or directory**

**cat link2**

получаем - **aaa**

Снова создадим файл с именем file\_a, но с другим содержимым:

**echo "bbb" > file\_a**

А теперь снова попробуем достучаться по ссылкам. Первая ссылка снова работает, и по ней будет находиться файл с содержимым «bbb». А по второй ссылке находится всё тот же файл с содержимым «aaa».

**cat link1**

получаем - **bbb**

**cat link2**

получаем - **aaa**

**Задание**

1. Войдите в систему с учетной записью **Student**.
2. Выведите на экран список файлов текущего каталога в краткой и расширенной форме.
3. Сделать каталог **/** текущим.
4. Сохранить в файле **$HOME/filelist.txt** список каталогов в каталоге /. **($HOME** - переменная окружения, которая содержит путь к «домашнему» каталогу пользователя).
5. Вернитесь в домашний каталог и выведите рекурсивный список всех файлов и каталогов в каталоге **/usr/ports/www**.
6. В домашнем каталоге создайте подкаталоги **src**, **dst**, **temp**.
7. В каталоге **src** создайте текстовый файл **f1** произвольного содержания.
8. В каталог **src** скопируйте файлы **info.txt**, **ps.txt** и **ps1.txt**, созданные в предыдущей работе. Все ли файла получилось скопировать?
9. В каталоге **dst** создайте «жесткие» ссылки на все файлы из каталоге **src.**
10. В домашнем каталоге создайте «мягкие» ссылки на файлы из каталоге **src**.
11. Выведите рекурсивно расширенную информацию о содержимом домашнего каталога. Обратите внимание на размер для физических файлов и ссылок. Результат сохраните в файл в домашнем каталоге.
12. Находясь в домашнем каталоге выполнить команды и запомните результат:

* **Cat src/f1**
* **Cat dst/f1\_hard\_link.txt**
* **Cat f1\_soft\_link.txt**

1. Переместите файл **f1** из каталога **src** в каталог **temp** и повторите пункт 12.
2. Удалите файл **f1** и повторите пункт 12.
3. Удалите все файлы с расширением **.txt** из каталога **dst**.
4. Удалите каталог **dst.**
5. Переместите каталог **temp** в **src.**
6. Выведите рекурсивно расширенную информацию о содержимом домашнего каталога. Обратите внимание на размер для физических файлов и ссылок. Результат сохраните в файл в домашнем каталоге.
7. Рекурсивно удалите каталог **src.**
8. Завершите сеанс.
9. Войдите в систему с учетной записью **root**.
10. Выполните команду **df –h**. Обратите внимание на первый и последний столбцы результата. Первый – это наименование устройства, соответствующее дискам, а второй это точка монтирование файловой системы. Нас будет интересовать второй диск, на который мы при инсталляции установили порты.
11. С помощью команды **umount** отмонтируйте файловую систему второго (меньшего, 5Гб) диска.
12. Посмотрите результат с помощью команды **df**.
13. С помощью команды **mount** восстановите первоначальное состояние – примонтируйте диск в **/usr/ports**.
14. Завершите сеанс и выключите виртуальную машину.

# Лабораторная работа 5 Работа в ОС FreeBSD.Пользователи, группы. Управление правами доступа.

**Цель**

Изучить основные команды управления группами и пользователями в ОС FreeBSD. Освоить концепцию управления правами доступа к файлам и каталогам.

**Теоретические сведения**

**Пользователи**

Весь доступ к системе осуществляется через учетные записи, и все процессы запускаются пользователями.

С каждой учетной записью в системе FreeBSD связана определенная идентификационная информация:

* **Имя пользователя.** Имя пользователя в том виде, в каком оно вводится в приглашение **login:**. Имена пользователей должны быть уникальны в пределах одного компьютера; не может быть двух пользователей с одинаковым именем пользователя.
* **Пароль.** С каждой учетной записью связан пароль. Пароль может быть пустым, в этом случае для доступа к системе не нужен пароль.
* **ID пользователя** (User ID, UID). Это номер от 0 до 65535, используемый для однозначной идентификации пользователя в системе. Сама система FreeBSD для идентификации пользователей использует UID – любая команда FreeBSD, позволяющая указывать имя пользователя, первым делом преобразует его к UID. Теоретически, можно создать несколько учетных записей с различными именами пользователей, но с одним UID.
* **ID группы** (Group ID, GID). Это номер от 0 до 65535, используемый для однозначной идентификации главной группы, к которой принадлежит пользователь. Группы это механизм для контроля доступа к ресурсам на основе GID пользователя вместо его UID. Пользователь может быть включен более чем в одну группу.
* **Класс логина.** Это расширение к механизму групп, позволяющее системе более гибко управлять различными пользователями.
* **Время изменения пароля.** По умолчанию FreeBSD не принуждает пользователей периодически менять пароли.
* **Время истечения действия учетной записи.** По умолчанию в FreeBSD время действия учетных записей не ограничено. После наступления этого времени учетная запись не может использоваться для входа в систему, хотя каталоги и файлы этой учетной записи останутся.
* **Полное имя пользователя.** Имя пользователя является уникальным идентификатором учетной записи в FreeBSD, но недостаточно для сопоставления с реальным именем пользователя. Эта информация может быть добавлена в учетную запись.
* **Домашний каталог. Э**то полный путь к каталогу в системе, в котором пользователь начнет работать после входа в систему. По общепринятому соглашению все домашние каталоги пользователей помещаются в **/home/username** или **/usr/home/username**.
* **Оболочка пользователя.** Необходима пользователям как средство взаимодействия с системой.

Существует три основных типа учетных записей:

* **Cуперпользователь** – **root** Используется для управления системой без ограничения привилегий. Существует в системе изначально. Не рекомендуется использовать для повседневных задач.
* Системные пользователи. Предназначены для запуска сервисов. Это сделано по соображениям безопасности, что бы не запускать сервисы от имени root. Например, **daemon, operator, www, bind**. **nobody** – это классический непривилегированный пользователь.
* **Учетные записи пользователей.** Позволяют обычным пользователям получить доступ к системе и позволяют настроить для каждого персонально рабочую среду.

В системе FreeBSD используется технология «теневых» паролей – системные данные пользователей разнесены по двум файлам:

* **/etc/master.passwd** – находятся пароли в зашифрованном виде, он имеет права только на чтение и запись для пользователя root
* **/etc/passwd** – созданный с помощью команды **pwd\_mkdb** (генерация базы с паролями) из файла **/etc/master.passwd**, он имеет права на чтение для группы и других пользователей, в нём пароли заменены на \*. Кроме этого, с помощью команды **pwd\_mkdb** из файла **/etc/master.passwd** создаются два файла – **/etc/pwd.db** и **/etc/spwd.db** (индексированные базы данных), они предназначены для ускорения поиска, в случае большого количества системных пользователей.

Для быстрого редактирования файла **/etc/master.passwd** и без последующего применения команды **pwd\_mkdb** применяется команда **vipw**, это тот же редактор **vi**. При использовании **\*** в **/etc/master.passwd** в поле **password** запрещается авторизация в системе так как символ \* не может быть зашифрованным паролем. Например, для временной блокировки пользователя, можно использовать в поле **password** в файле **/etc/master.passwd** такую комбинацию **\*LOCKED\*** или добавить такую комбинацию в начало пароля, если он существует, а при разблокировке просто её удалить. Если Вы хотите запретить доступ пользователя к системе можно заменить его оболочку на **/sbin/nologin**.

**Управление ресурсами пользователя и ограничения.**

Управление ресурсами пользователя осуществляется с помощью классов, которые определяются в специальном файле **/etc/login.conf**, а также задаются пользователю при его добавлении. Если для пользователя не определён какой-либо класс, то ему присваивается класс – **default**. Каждый класс имеет набор характеристик в виде **имя=значение**. Для ускорения доступа к данным система непосредственно не читает файл **/etc/login.conf**, а вместо этого читает файл **/etc/login.conf.db**, который создаётся специальной командой **cap\_mkdb**.

**Группы**

Группа это просто список пользователей. Группа идентифицируется по имени и **GID** (Group ID, идентификатор группы). В FreeBSD ядро для определения прав процесса использует два фактора: его ID пользователя и список групп, которым он принадлежит.

Имена групп связываются с ID групп в файле **/etc/group**. Это текстовый файл с четырьмя разделенными двоеточием полями:

* первое поле это имя группы;
* второе это зашифрованный пароль;
* третье это ID группы,
* четвертое это разделенный запятыми список членов группы.

Для редактирования **/etc/group** можно использовать команду [**pw**](http://www.freebsd.org/cgi/man.cgi?query=pw&amp;sektion=8) для добавления и редактирования групп.

**Команды для управления пользователями.**

* **last** – показывает, когда пользователь последний раз пользовался терминалом;
* **ac** – показывает время, которое пользователь находился в системе;
* **sa** – показывает статистику по пользователям;
* **adduser** – добавление нового пользователя. Создает записи в файлах **passwd** и **group**, создает домашний каталог пользователя.
* **rmuser** – удаляет пользователя из системы в интерактивном режиме.
* **chpass** – изменяет информацию в базе данных пользователей: пароль, оболочку, персональную информацию.
* **passwd** – изменяет собственный пароль пользователя или другого пользователя.
* **pw** – утилита командной строки для управления пользователями и группами

Для полноценного использования команды **sa** необходимо включить учёт используемых ресурсов. Для того что бы включить систему учёта используемых ресурсов необходимо выполнить следующие действия:

* mkdir /var/account
* touch /var/account/acct
* accton /var/account/acct
* echo 'accounting\_enable="YES"' >> /etc/rc.conf

Непосредственно систему учета включает команда **accotn**.

Добавляем нового пользователя, использую утилиту **pw**:

**pw useradd test -s /bin/sh -c "Test users" -m -b /home -e 03-06-2012 -p 02-06-2012**

* **-s** – указывает, какая оболочка будет использоваться
* **-с** – комментарии к созданному пользователю
* **-e** – время жизни учетной записи. Формат задания даты или времени таков: dd-mm-yy[yy], где dd – день, mm – месяц, yy[yy] – год.
* **-p** – время жизни пароля
* **-m** – заставляет создать домашний каталог пользователя и скопировать в него стандартные файлы из каталога /usr/share/skel
* **-b** – базовая директория в которой будет находиться домашний каталог пользователя
* **-L** – задаёт класс для пользователя из файла login.conf.

Добавляем новую группу и добавляем в нее пользователя test:

**echo 'testGroup:\*:200:test' >> /etc/group**

Создаем новую группу, добавляем в нее пользователей test и Student. Смотрим созданную группу:

**pw groupadd testGroup2 -M test,root**

**pw groupshow testGroup2**

**Управление правами доступа**

Существует три типа прав доступа:

* чтение;
* запись;
* исполнение.

Права сгруппированы три по три, соответственно чтение/запись/выполнение для владельца/группы/всех остальных. Численное представление:

|  |  |  |
| --- | --- | --- |
| **Значение** | **Права доступа** | **Символьное представление** |
| 0 | Ничего не разрешено | --- |
| 1 | Нельзя читать и писать, разрешено исполнять | --x |
| 2 | Нельзя читать и исполнять, разрешено писать | -w- |
| 3 | Нельзя читать, разрешено писать и исполнять | -wx |
| 4 | Разрешено читать, нельзя писать и исполнять | r-- |
| 5 | Разрешено читать и исполнять, нельзя писать | r-x |
| 6 | Разрешено читать и писать, нельзя исполнять | rw- |
| 7 | Разрешено все | rwx |

Права на устройства контролируются аналогичным образом. В FreeBSD все устройства представлены в виде файлов, которые можно открывать, читать и писать в них. Эти специальные файлы содержатся в каталоге **/dev**. Каталоги также являются файлами. К ним применимы те же права на чтение, запись и выполнение. В данном случае «выполнение» имеет несколько другой смысл. Когда каталог помечен как «исполнимый», это означает, что можно «зайти» в него (с помощью команды cd). Это также означает, что в данном каталоге можно получить доступ к файлам, имена которых известны (конечно, если собственные права на файл разрешают такой доступ). Если же требуется получить список файлов в некотором каталоге, права доступа на него должные включать доступ на **чтение**. Для того, чтобы удалить из каталога какой-либо файл, имя которого известно, на этот каталог должны быть даны права **на запись и на исполнение**.

**Символические обозначения прав**

Используются вместо числовых обозначений прав.

|  |  |  |
| --- | --- | --- |
| **Опция** | **Буква** | **Значение** |
| (кто) | u | Пользователь (User) |
| (кто) | g | Группа (Group) |
| (кто) | o | Другие (Other) |
| (кто) | a | Все (All, ''world'') |
| (действие) | + | Добавление прав |
| (действие) | - | Удаление прав |
| (действие) | = | Явная установка прав |
| (права) | r | Чтение (Read) |
| (права) | w | Запись (Write) |
| (права) | x | Выполнение (Execute) |

**Команда управления правами доступа**

**сhmod** – изменения прав доступа к ресурсу;

**chown** – изменение группы и владельца ресурса;

**chgrp** – изменение группы ресурса;

**umask** – задает маску на создаваемые файлы для пользователя; Задаются три цифры, которые затем поочерёдно вычитаются из 7, если команда определяет права для каталогов и из 6, если команда определяет права для файлов.

**stat** – получение статистической информации о файле.

Примеры использования **chmod**

|  |  |
| --- | --- |
| chmod go+w hello.txt | разрешить запись для группы и прочих |
| chmod ug+x hello.txt | разрешить выполнение для владельца и группы |
| chmod a-x hello.txt | запретить выполнение для всех (a == ugo) |
| chmod go-w hello.txt | запретить запись для группы и прочих |
| chmod 755 hello.txt | разрешить чтение и выполнение всем и запись владельцу |
| chmod 644 hello.txt | запретить выполнение всем |
| chmod 711 hello.txt | разрешить только выполнение для группы и прочих |

**Задание**

1. Войдите в систему с учетной записью **root.**
2. Создайте пользователя **Student2** с помощью команды **adduser** в интерактивном режиме.
3. Посмотрите содержимое файла **/etc/master.passwd**. Убедитесь, что пользователя действительно добавили.
4. Измените пароль у пользователя **Student2**.
5. Создайте нового пользователя **Student3** с помощью утилиты **pw**, при этом в качестве оболочки должен использоваться **/bin/csh**; директория, где будет находиться домашний каталог – **/var/tmp**; время действия пароля – текущая дата плюс неделя.
6. Создайте новую группу **Students** и включите в нее всех студентов (**Student, Student2, Student3**).
7. Посмотрите информацию по созданной группе (**Students**).
8. Удалите пользователя **Student2.** Предварительно в текстовый файл вывести информацию по учетным записям системы до удаления и после.
9. Завершить все открытые сеансы.
10. Войти в систему под учетной записью **Student3**.
11. Создать в домашнем каталоге 2-3 файла произвольного содержания (имена файлов - **u1, u2, u3**).
12. Повторить пункт 10 от имени пользователя root в новом сеансе (имена файлов - **r1, r2, r3**; должны находиться в домашнем каталоге пользователя **Student3**).
13. Завершить сеанс **root.**
14. Переключиться в **ttyv0.**
15. Получить развернутый список файлов домашнего каталога пользователя **Student3** и сохранить его в файле **listing1**
16. Просмотреть файл **listing1**, обратив внимание на поля прав доступа, владельца и группы.
17. В **ttyv2** открыть сеанс для пользователя **root**.
18. Перейти в домашний каталог пользователя **Student3**.
19. Изменить права доступа к файлам **u1** и **r1** следующим образом:

**u1** – запретить запись для владельца и группы

**r1** – разрешить запись для всех

1. Переключиться в **ttyv0** и изменить содержимое файлов **u1** и **r1**. Сохранить изменения.
2. Перейти в **ttyv2** и изменить владельца файлов **u1** и **u2** на **root**, а группу на **Students**
3. Перейти в **ttyv0** и попробовать изменить файл **u2**
4. Создать файл **hello** следующего содержания:

**#! /bin/sh**

**echo “Hello, World!”**

**echo –n “Privet!”**

**whoami**

1. Выполнить следующие действия и проанализировать результат

* набрать в командной строке имя файла **hello** и нажать Enter
* набрать в командной строке **sh hello** и нажать Enter
* установить для файла **hello** права на **выполнение** для всех, набрать в командной строке **./hello** и нажать Enter

1. В терминале **ttyv2** создайте каталоги **/home/shared, home/shared/pub, /home/shared/upload, /home/shared/temp**. Установить на них следующие права:

|  |  |  |  |
| --- | --- | --- | --- |
| **каталог** | **владелец** | **группа** | **права** |
| pub | root | wheel | 775 |
| upload | nobody | nobody | 130 |
| temp | Student | Students | 777 |

1. Самостоятельно изучите информацию по теме «Списки кортроля доступа файловой системы (ACL)» (<https://www.freebsd.org/doc/ru_RU.KOI8-R/books/handbook/fs-acl.html>). Реализуйте возможный сценарий использования ACL по своему усмотрению для наилучшей демонстрации возможностей и ACL в ОС FreeBSD.

# Лабораторная работа 6 Первоначальная настройка сервера на основе FreeBSD 11.1. Текстовый редактор «ee». Команда su. Утилита sudo. Установка bash. Локализация консоли. Настройка ssh, ftp.

**Цель**

Изучить возможности текстового редактора **ee**. Научиться выполнять первоначальную настройку сервера на основе FreeBSD 11.1.

**Теоретические сведения**

**Редактор ee**

Самым простым в изучении и использовании, можно назвать **ee**, что расшифровывается как «easy editor», т.е. «простой редактор». Чтобы начать редактировать какой-либо файл, наберите в командной строке **ee filename**, где **filename** имя редактируемого файла. Например, для редактирования файла **/etc/rc.conf**, наберите **ee /etc/rc.conf**. В верхней части экрана находится список основных команд редактора. символ каретки (^) означает клавишу Ctrl. Чтобы выйти из редактора, нажмите клавишу Esc, затем Enter. Если остались какие-либо несохраненные данные, то потребуется подтвердить выход, либо сохранив результат работы, либо нет.

FreeBSD также поставляется с мощным текстовым редактором **vi**, как часть системы, и редакторами **emacs и vim**, которые можно найти в коллекции портов.

**Команда su**

Команда **su** позволяет пользователю выполнять команды от имени другого пользователя, не завершая текущий сеанс, или получить роль. По умолчанию предполагается работа от имени пользователя **root** (суперпользователя).

Для использования **su** необходимо ввести соответствующий пароль (если только команду не вызывает пользователь root). Если введен правильный пароль, **su** создает новый процесс командного интерпретатора, с такими же реальными и эффективными идентификаторами пользователя и группы, а также списком дополнительных групп, что и у указанного пользователя.

**Пользователь для выполнения su должен быть членом группы wheel.**

**Синтаксис:**

su [-] [имя\_пользователя [аргумент ... ]]

**Утилита sudo**

Утилита ***sudo***используется для замены стандартному **su**, и позволяет получить права пользователя ***root***без указания его пароля.

В конфигурационном файле **sudo** **(/usr/local/etc/sudoers**) можно указывать логин юзера, список команд, которые ему доступны и спрашивать ли его пароль при запуске какой либо команды через **sudo**.

**Командный интерпретатор**

При работе с FreeBSD, для выполнения повседневных задач используется командный интерфейс (так называемая «оболочка», «shell»). Основная задача интерпретатора – принимать вводимые команды и выполнять их. Вместе с FreeBSD поставляется несколько командных интерпретаторов, например, sh, или Bourne Shell, и tcsh, расширенная версия C-shell. Многие другие интерпретаторы доступны из коллекции портов FreeBSD, например zsh и bash.

Одна из наиболее часто используемых функций командного интерпретатора – дополнение частичного имени файла до полного. Вы можете набрать только первые несколько символов имени файла, нажать клавишу табуляции (TAB), и командный интерпретатор автоматически завершит имя.

Дополнительные возможности при работе с интерпретатором дает использование переменных окружения. **Переменные окружения** – это пары переменная/значение, хранящиеся в памяти интерпретатора. Значение переменных окружения может быть прочитано любой программой, запущенной из командного интерпретатора, и часто содержит настройки для многих приложений и утилит. Ниже приведены некоторые наиболее часто встречающиеся переменные окружения и их значения:

| **Переменная** | **Описание** |
| --- | --- |
| USER | Имя текущего пользователя. |
| PATH | Каталоги, разделенные двоеточием, для поиска исполняемых файлов. |
| DISPLAY | Сетевое имя виртуального дисплея X11, доступного для подключения. |
| SHELL | Текущий командный интерпретатор. |
| TERM | Тип терминала пользователя. Используется, чтобы узнать возможности терминала. |
| TERMCAP | Список escape-последовательностей для управления различными функциями терминала. |
| OSTYPE | Название (тип) операционной системы. Например, FreeBSD. |
| MACHTYPE | Архитектура машины (процессора). |
| EDITOR | Выбранный пользователем текстовый редактор. |
| PAGER | Выбранная пользователем утилита просмотра файлов. |
| MANPATH | Каталоги, разделенные двоеточием, для поиска файлов системного справочника. |

Установка значений переменных окружения различна для разных оболочек. Например, в интерпретаторах C-стиля, таких как **tcsh** и **csh**, это **setenv**. В интерпретаторах Bourne, таких как **sh** и **bash**, это **export**. Например, чтобы установить или изменить значение переменной **EDITOR** к значению **/usr/local/bin/emacs** в **csh** или **tcsh**, выполните команду:

**setenv EDITOR /usr/local/bin/emacs**

В оболочках Bourne:

**export EDITOR="/usr/local/bin/emacs"**

Чтобы получить значение переменной, например, в командной строке, поместите символ **$** перед именем переменной. Например, команда **echo $TERM** выведет значение переменной **$TERM**.

**Протокол ssh**

**SSH** является полноценным сетевым протоколом для безопасной передачи данных и обязательным инструментом любого администратора Unix серверов.

Настроек **SSH**по умолчанию, вполне достаточно для решения большинства задач. Перенастройка **SSH**сервера может потребоваться в случае необходимости воспользоваться каким-то дополнительным функционалом, не включенным по-умолчанию.

[Настройка сервера](http://vds-admin.ru/) **SSH**, то есть программы-демона **sshd**, производится через файл конфигурации, расположенный по адресу: ***/etc/ssh/sshd\_config***.

**Служба FTP. Протокол FTP.**

**Служба FTP** (от протокола - File Transfer Protocol) – предназначена для обмена файлами. FTP служба построена по схеме «клиент-сервер». Клиент (браузер,Windows Commander...) посылает запросы серверу и принимает файлы. Сервер FTP (ftpd, proftpd…) обрабатывает запросы клиента на получение файла.

Служба FTP базируется на двух стандартах:

* URL (Universal Resource Locator) – универсальный способ адресации ресурсов в сети ;
* FTP (File Transfer Protocol) – протокол передачи файлов.

**File Transfer Protocol** – протокол передачи файлов, протокол высокого уровня (уровня приложений), используется службой FTP для передачи файлов. FTP отличается от других приложений тем, что он использует два TCP соединения для передачи файла.

* Управляющее соединение – соединение для посылки команд серверу и получение ответов от него. (порт 21)
* Соединение данных – соединение для передачи файлов (порт 20).

**Программа ftpd**

Это встроенный в FreeBSD ftp-сервер.

Флаги запуска демона:

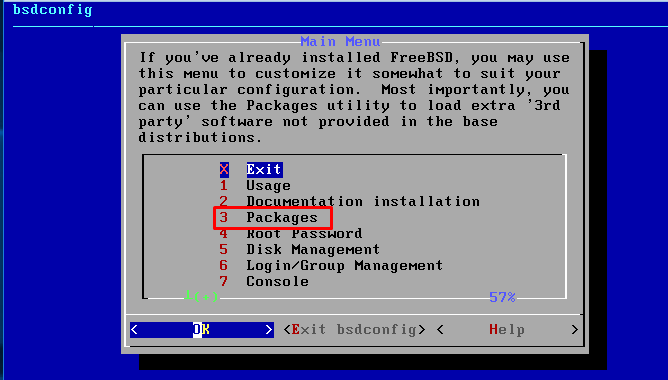
* **-D** – запускать ftpd в режиме демона.
* **-a <адрес>** – в режиме демона принимать соединения только на указанный IP-адрес.
* **-d** – включить режим отладки (подробная информация о работе будет выдаваться серверу syslog как LOG\_FTP).
* **-h** – не выводить информацию о системе в сообщениях сервера.
* **-l** – протоколировать все сессии (обычно используется/var/log/xferlog).
* **-A** – разрешить только анонимный доступ.
* **-M** – запретить анонимным пользователям создавать папки.
* **-m** – разрешить анонимным пользователям модифицировать существующие файлы (если для этого достаточно системных прав).
* **-o, -O** – разрешить только запись на сервер всем пользователям (-o) или только анонимному пользователю (-O). В частности, таким образом можно организовать сбор с удаленных серверов файлов резервных копий, чтение которых пользователями не предусмотрено.
* **-r** – перевести сервер в режим «только для чтения». Любая модификация размещенных данных будет запрещена.

Конфигурационные файлы, входящие в состав демона:

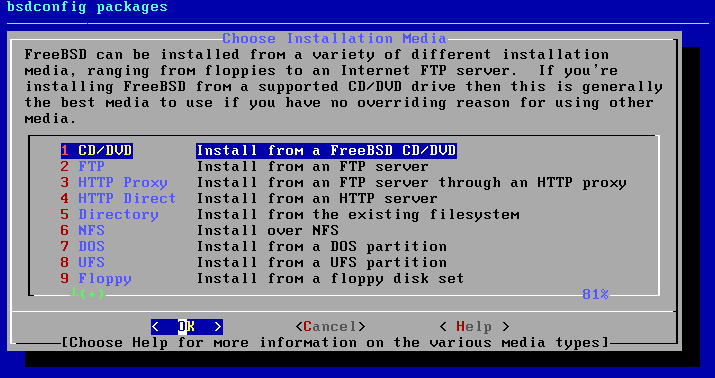
* **/etc/ftpusers** – список запрещенных пользователей.
* **/etc/ftphosts** – конфигурация виртуального хостинга.
* **/etc/ftpwelcome** – приветствие нашего сервера
* **/etc/ftpmotd** – сообщение после логина
* **/var/run/ftpd.pid** – файл, содержащий номер процесса (pid) для работы в режиме демона.
* **/var/run/nologin** – сообщение о запрете доступа.
* **/var/log/ftpd** – лог о действиях анонимных пользователей.
* **/etc/ftpchroot** – список пользователей и групп, к которым применяется ограничение доступа.

**Выполнение**

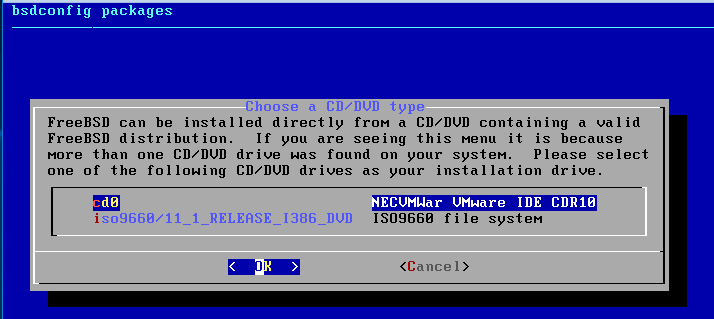
1. Войдите в систему под учетной записью **Student**.
2. Используя редактор **ee**, создайте файл произвольного содержания с именем **test.txt**.
3. Попробуйте основные команды редактора **ee**.
4. Откройте новый сеанс в терминале под учетной записью **root**.
5. Создайте текстовый файл (**test\_su.txt**) в домашнем каталоге суперпользователя с помощью редактора **ee**.
6. Перейдите в терминал **ttyv0** (там работаем под **Student**).
7. Перейдите в домашний каталог суперпользователя и попытайтесь изменить содержание файла **test\_su.txt**. Объясните, почему не получается.
8. Выполните команду **su**. Введите пароль суперпользователя.
9. Попробуйте еще раз выполнить пункт 7. Объясните, почему получилось.
10. Выполните команду **exit**.
11. Откройте новый сеанс в терминале **ttyv4** под учетной записью **Student3**.
12. Создайте текстовый файл (**test\_su.txt**) в домашнем каталоге пользователя **Student3** с помощью редактора **ee**.
13. Перейдите в терминал **ttyv0** (там работаем под **Student**).
14. Перейдите в домашний каталог **Student3** и попытайтесь изменить содержание файла **test\_su.txt**. Объясните, почему не получается.
15. Выполните команду **su –s Student3**. Введите пароль.
16. Попробуйте еще раз выполнить пункт 14. Объясните, почему получилось.
17. Выполните команду **exit**.
18. Выполним установку утилиты **sudo** с диска в виде бинарного пакета. (Проверьте: установочный диск должен быть подключен к виртуальной машине. Используйте диск **FreeBSD-11.1-RELEASE-i386-dvd1.iso**).
19. Выполните команду **su** для получения прав суперпользователя.
20. Выполните запуск утилиты **bsdconfig**.



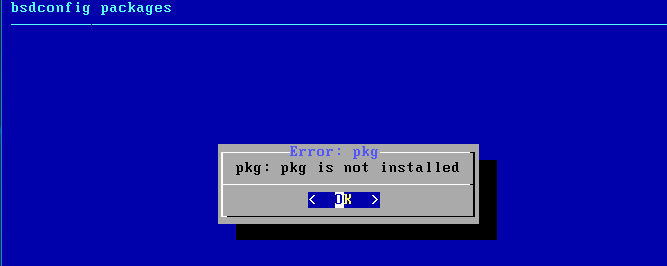
1. Выбираем пункт **Packages** и затем **CD/DVD**.



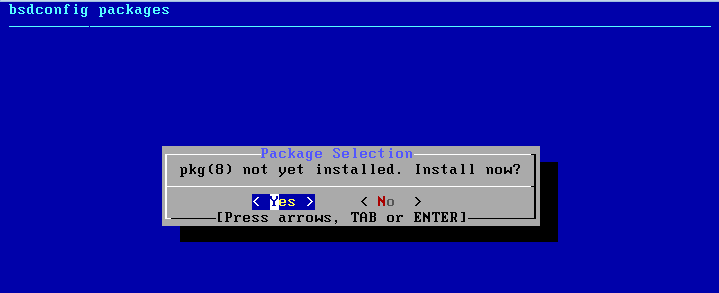
1. Выбираем привод на котором «предположительно» ☺ находятся дистрибутивы программ.



1. Получаем сообщение, что не установлена программа **pkg**.



1. Система предлагает нам установить данный пакет, соглашаемся. Если правильно была выполнена настройка сетевых параметров и связь с серверами ftp работает, то будет выполнено скачивание и установка пакета.



1. Небольшое замечание: если Вы работаете через прокси сервер, то необходимо настроить дополнительные параметры

- в файлы **/.cshrc** или **/home/%username%/.cshrc** (папка Вашего пользователя) добавляем строки следующего формата

ee

- в файл **/etc/make.conf** добавляем строки следующего формата

FETCH\_ENV=HTTP\_PROXY=http://login:pass@server:port

FETCH\_ENV=FTP\_PROXY=http://login:pass@server:port

FETCH\_ENV=http\_proxy=http://login:pass@server:port

FETCH\_ENV=ftp\_proxy=http://login:pass@server:port

1. После успешной установки получаем сообщение, которое по-сути означент, что не найден список пакетов. Придется системе указать, где располагаются пакеты для установки, но, не смотря на все препятствия, теперь мы можем пользоваться программой **pkg** для установки бинарный пакетов.

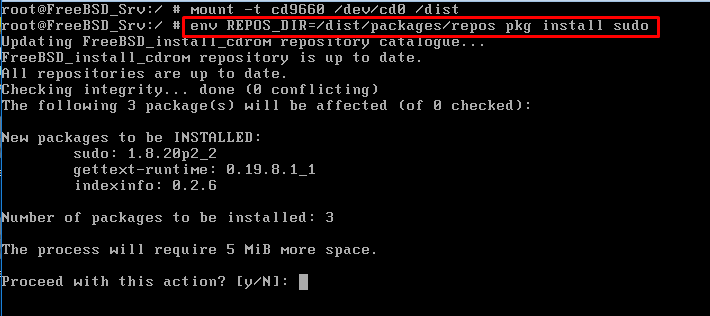


1. Выполним следующую команду и убедимся, что **Package manager**(**pkg)** действительно установлен в системе.

**pkg info**

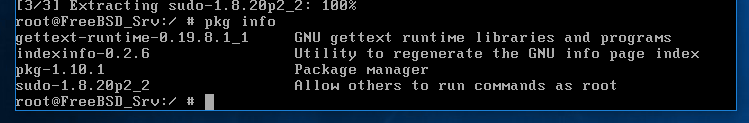


1. Продолжаем пытаться установить программу **sudo** :). Подмонтируем в каталог **/dist** папку с дистрибутивами установочного диска (для Вас уже это не составляет труда). После этого указав системе расположение дистрибутивов через переменную среды **REPOS\_DIR** выполним установку пакета **sudo**.



1. C помощью известной команды убеждаемся, что пакет sudo и связанные с ним зависимости установлены.

**pkg info**



1. В файле **/usr/local/etc/sudoers** разрешим пользователю **Student** выполнять команды от имени суперпользователя:

**## User privilege specification**

**##**

**root ALL=(ALL) ALL**

**Student ALL=(ALL) ALL**

1. Завершите работу под учетной записью **root** с помощью команды **exit**.
2. Введите команду **sudo ee /root/test\_su.txt**, пробуем изменить файл, который принадлежит **root**. Команда **sudo** запросит пароль, вводим **СВОЙ** пароль (т.е. пользователя **Student**). Почуствуйте разницу между **sudo** и **su**. При использовании **sudo** нет необходимости знать пароль **root**.
3. Введите команду **sudo –s**.
4. Выполните команду **whoami**. Убедитесь, что стали **root**.
5. Выполните команду **exit**, затем **whoami**. Посмотрите на результат.
6. Установите интерпретатор **bash** в виде пакета, используя Packege manager. Можно попробовать найти такой пакет с помощью команды

**pkg search bash**

1. По совету системы выполните команду

**pkg update**

1. После установки убедитесь, что все закончилось успешно – выполните команду **pkg info**.
2. Замените интерпретатор («оболочку») для **root** на **bash** (**/usr/local/bin/bash**) и класс для пользователя на **russian**, вспомните несколько способов как это можно сделать.
3. Изменения вступят в силу после завершения сеанса и повторного входа в систему.
4. Проверьте текущий командный интерпретатор, выполните команду

**echo $SHELL**

1. Выполним локализацию консоли и немного упростим жизнь себе.
2. В файл **/boot/loader.conf** добавьтье следующую строку

**kern.vty=sc**

Начиная с 10 релиза, в FreeBSD присутствуют одновременно **device sc** и **device vt**. Это два разных драйвера консоли, более старый - **sc**, более современный - **vt**. по умолчанию в 10+ используется **vt**. Нам придется использовать более старый, т.к. поддержка нового не реализована на уровне vmware (впрочем как и в hyper-v).

1. Отредактируейт файл /etc/login.conf следующим образом:

**#russian|Russian Users Accounts:\**

**# :charset=UTF-8:\**

**# :lang=ru\_RU.UTF-8:\**

**# :tc=default:**

**russian|Russian Users Accounts:\**

**:charset=KOI8-R:\**

**:lang=ru\_RU.KOI8-R:\**

**:tc=default:**

Тем самым мы создали новый профиль для логина.

1. Выполните команду

**cap\_mkdb /etc/login.conf**

1. Отредактируйте файл **/etc/rc.conf** следующим образом:

**font8x14="koi8-r-8x14"**

**font8x16="koi8-r-8x16"**

**font8x8="koi8-r-8x8"**

**keymap="ru.koi8-r.win"**

**keyrate="fast"**

Для переключения между раскладками использовать теперь будем комбинации клавиш **CTRL+SHIFT**.

1. В файлы **/etc/profile** и **.profile** в домашних каталогах пользователей добавьте следующие строки:

**export LANG=ru\_RU.KOI8-R**

**export LC\_ALL=ru\_RU.KOI8-R**

**export EDITOR=ee**

1. Проверьте содержимое файла **/etc/ttys** там должна быть примерно следующая строка для **ttyv0** ну и для остальных терминалов. Важным здесь является **cons25r**.

**ttyv0 "/usr/libexec/getty Pc" cons25r on secure**

1. Выполните перезагрузку системы, что бы изменения вступили в силу.
2. Для проверки произведенных действий можно открыть новый сеанс и зайти под учетной записью **root**.
3. Измените «оболочку» на **bash** и класс пользователя на **russian** для **Student** с помощью команды **vipw**. Проверьте результативность действий.
4. Завершите сеанс и войдите снова в систему как **root**. Проверьте возможность вводить русские буквы.
5. К сожалению редактор ee не поддерживает «русские буквы» в UTF-8, поэтому можно установить редатор vim.
6. Отредактируем файл **/etc/rc.conf** для запуска демона **ftp**. Добавляем строчку **ftpd\_enable=”YES”**

**ftpd\_flags=”-D –l”**

1. Запускаем процесс вручную, чтобы не перезагружаться

**/etc/rc.d/ftpd start**

1. Проверяем ожидает ли систма подключения к ftp серверу

**sockstat –v | grep :21**

1. Убедитесь, что можно подключиться к ftp серверу. Попробуйте подключиться с помощью любого ftp клиента из базовой ОС. В состоянии подключения выполните команду

**sockstat –v | grep :21**

1. Для обеспечения возможности подключения с помощью протокола ssh, отредактируйте файл /etc/rc.conf следующим образом:

**sshd\_enable=”YES”**

1. Запустите демона вручную командой

**/etc/rc.d/sshd start**

1. Проверяем ожидает ли систма подключения по протоколу ssh

**sockstat –v | grep :22**

1. Детальная настройка демона **sshd** выполняется с посмщью файла **/etc/ssh/sshd\_config**
2. В качестве ssh клиента будем использовать программу **PuTTY** (можно скачать русскую версию). Она не требует инсталляции и предоставляет достаточно возможностей для настройки подключения.
3. Запустите PuTTY. В секции **Session** указываем имя нашего сервера или его IP адрес (192.168.180.2). В секции **Windows -> Translation** в раскрывающемся списке **Remote character set** выбираем **KOI8-RU**. В секции **Connection ->Data** в поле **Auto-login username** указываем **Student** (что бы каждый раз не вводить имя пользователя). Переходим в секцию **Session** и в поле **Saved Sessions** указываем произвольное имя для сохранения настроек, например, Server\_FreeBSD. Нажимаем кнопку **Save**.
4. Закройте программу PuTTY
5. Запустите программу PuTTY.
6. В секции **Session** в списке сохраненных сессий выберите сохраненную в предыдущем пункте сессию и нажмите **Load**, тем самым загрузим настройки.
7. Нажмите кнопку **Open**. Если все сделали правильно, то достаточно будет ввести пароль и откроется сеанс работы с системой.
8. Для завершения сеанса введите команду **exit**.
9. Настроим подключение к системе с авторизацией по ключам.
10. Запустите программу PuTTY и откройте сеанс под учетной записью **Student**.
11. Выполним команду **ssh-keygen**.
12. Система предложит указать путь для хранения ключа, согласимся использовать его (это домашний каталог пользователя, **.ssh/id\_rsa**)
13. Указываем пароль для ключа. Повторно вводим заданный пароль.
14. В результате система сообщает, что было сгенерировано два ключа: **id\_rsa** – private key и **id\_rsa.pub** – public key.
15. Выполним команду **ls –lah** и убедимся, что каталог **.ssh** в домашнем каталоге пользователя создан.
16. Переходим в каталог **.ssh** и смотрим его содержимое, там должно находиться два файла с ключами.
17. Добавим содержимое файла **id\_rsa.pub** в конец файла **authorized\_keys**

**cat id\_rsa.pub >> authorized\_keys**

1. Если файла **authorized\_keys** еще не существовало, то можно его создать простым копированием **id\_rsa.pub.**

**cp id\_rsa.pub authorized\_keys**

1. Проверяем наличие трех файлов в каталоге **.ssh**.
2. Файлы **id\_rsa** и **id\_rsa.pub** желательно убрать в другое место и не потерять. Пока оставьте все как есть.
3. Для авторизации с помощью ключей необходимо немного подправить конфигурационный файл демона sshd.
4. Откройте файл **/etc/ssh/sshd\_config** и раскомментируйте строки (к сожалению прав на изменение этого файла у пользователя **Student** нет, поэтому подумайте немного как это можно сделать в текущем сеансе):

**Protocol 2**

**RSAAuthentication yes**

**PubkeyAuthentication yes**

**AuthorizedKeysFile %h/.ssh/authorized\_keys**

**PasswordAuthentication no**

1. Выполняем перезапуск sshd

**sudo /etc/rc.d/sshd stop**

**sudo /etc/rc.d/sshd start**

**или**

**sudo /etc/rc.d/sshd restart**

1. Выполните команду **exit**.
2. С помощью **ftp** заберите private key (файл **id\_rsa**) с сервера, он понадобится для авторизации в программе PuTTY.
3. Единственно неудобство заключается в том, что придется преобразовывать private key, который сгенерировали на сервере в формат понятный PuTTY.
4. Запустите программу **Puttygen** на базовой станции.
5. Нажмите кнопку **Load**. Выберите скачанный с сервера private key – **id\_rsa** (при выборе файла измените фильтр по расширению). Введите пароль, который использовали при генерации ключа. Таким образом, ключ был успешно импортирован в программу Puttygen.
6. Нажмите кнопку **Save private key**. Укажите имя, под которым будет сохранен ключ в формате, понятным для PuTTY.
7. Закройте программу Puttygen.
8. Запустите программу PuTTY.
9. Загрузите настройки соединения.
10. В секции **Connection -> SSH -> Auth** в поле **Private key file for authentication** выберите преобразованный privete key.
11. Сохраните измененные настройки и выполните открытие сеанса. Вам придется ввести пароль, который был использован при генерации private key.
12. Настроим синхронизацию времени системы из сети
13. Редактируем файл **/etc/rc.conf**, добавляем строки

**ntpdate\_enable="YES"**

**ntpd\_enable="YES"**

1. В файле **/etc/ntp.conf** изменяем список серверов, с которых будем брать время (можно оставить и те, что в конфиге по умолчанию)

**server 0.ru.pool.ntp.org iburst maxpoll 9**

**server 1.ru.pool.ntp.org iburst maxpoll 9**

**server 2.ru.pool.ntp.org iburst maxpoll 9**

**server 3.ru.pool.ntp.org iburst maxpoll 9**

1. Запускаем процесс немедленной синхронизации времени

**/etc/rc.d/ntpdate start**

1. Запускаем процесс-демон, который будет отвечать за синхронизацию времени

**/etc/rc.d/ntpd onestart**

1. В списке процессов убеждаемся, что демон успешно запущен

**ps ax | grep ntpd**

1. Таким образом, получили локализованную консоль, поменяли интерпретатор, настроили возможность подключения по защищенному протоколу **ssh** и передавать файлы с помощью протокола **ftp**.

# Лабораторная работа 7 Настройка Web-сервера (Apache). Организация файлового сервера на основе Samba.

**Цель**

Изучить основные этапы установки и настройки web-сервера Apache и организации виртуального web-хостинга на платформе FreeBSD. Обеспечить пользователям Windows доступ к файлам, расположенным на сервере FreeBSВ с помощью Samba.

**Теоретические сведения**

**Apache HTTP сервер**

Это свободно распространяемый веб-сервер, является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare.

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.

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации:

* Конфигурация сервера **(/usr/local/etc/apache24/httpd.conf**).
* Конфигурация виртуального хоста (**/usr/local/etc/apache24/httpd.conf** c версии 2.4).
* Конфигурация уровня директории (**.htaccess**).

Имеет собственный язык конфигурационных файлов, основанный на блоках директив.

**Samba**

Эта программа позволяющая обращаться к сетевым дискам по протоколу SMB/CIFS из разных операционных систем.

Возможности SAMBA:

* обеспечение доступа к Unix-разделам Windows-клиентам;
* обеспечение доступа к Windows-разделам Unix-клиентам;
* совместное использование принтеров подключенных к Unix-системе Windows-клиентами;
* совместное использование принтеров подключенных к Windows-системе Unix-клиентами;
* средства авторизации и аутентификации.

Типы аутентификации:

* **ADS** – samba является членом домена Active Directory. Необходимо поставить Kerberos чтобы использовать этот режим.
* **Domain** – samba полагается на информацию от Primary или Backup Domain Controller. Поступающие к ней пароли и логины она направляет к контроллеру и ждет пока тот проверит, есть ли такой пользователь в домене или нет.
* **Server** – samba пытается проверить валидность пользователя направляя его учетные данные на проверку другому Samba-серверу. Если это невозможно, то пытается использовать режим доступа User.
* **Share** – свободный доступ на Samba-сервер. Пароль не запрашивается пока пользователь не попытается попасть в какую либо специфичную область файловой системы, на которую стоит ограниченный режим доступа.
* **User** – используется по умолчанию. Аутентификация происходит на уровне SAMBA-сервера. Нужно создать системную учетную запись, а потом связать ее с учетной записью SAMBA.

Ключевые файлы:

* **/etc/samba/smb.conf** – Главный конфигурационный файл и все настройки задаются здесь;
* **/etc/samba/smbusers** – Файл в котором можно «залинковать» Windows-пользователя на UNIX-пользователя, например чтобы понять при подключение Windows-пользователя кто он будет на SAMBA-сервере. Еще одно назначение – это линковка многих windows-пользователей на одного UNIX-пользователя, в этом случае все действия в системе будут производится от его лица и с его правами доступа;
* **/etc/samba/lmhosts** – Похож на /etc/hosts только занимается резолвингом NetBIOS имен;
* **/usr/bin/smbpasswd** – Программа создания учетных записей SAMBA.
  + -a добавляем пользователя (он уже должен быть в системе);
  + -e включаем его учетную запись (по умолчанию отключена);
  + -x удаляем пользователя из базы данных SAMBA (системный пользователь остается на месте);
* **/usr/bin/smbclient** – SAMBA-клиент, позволяет подключаться к SAMBA-ресурсам из командной строки (после того как подключитесь наберите help для вывода возможных действие, help <command> - более подробное описание команды);
* /**usr/bin/smbcontrol** – Позволяет отсылать команды smbd и nmbd;

**Выполнение**

1. Запустите программу **PuTTY**.
2. Откройте сеанс под учетной записью **Student**.
3. Выполните инсталляцию web-сервера Apache24 с помощью Package manager.
4. Посмотрите, какие дополнительные пакеты были установлены (**pkg info**).
5. Выполним минимальную начальную настройку. Редактируем файл **(/usr/local/etc/apache24/httpd.conf**

директива **Listen –** если сервер имеет несколько сетевых интерфейсов, то можно указать Apache «слушать» запросы только на определенном порту и по определенному IP адресу

**оставляем без изменений**

директива **ServerName** – устанавливает полное доменное имя с использованием, которого производится обращение к web-серверу.

**указываем – www.ServerFreeBSD.MyCompany.ru**

1. Работая в хостовой ОС, проверяем доступность сервера FreeBSВ по IP-адресу (192.168.180.2), а затем по имени [www.ServerFreeBSD.MyCompany.ru](http://www.ServerFreeBSD.MyCompany.ru) с помощью команды ping. Если сервер не доступен по IP-адресу, то проверяем настройки сетевых подключений. Если не доступен по имени, то в файле **hosts** (для Windows XP – **C:\WINDOWS\system32\drivers\etc\hosts**) укажите соответствие IP адреса и имени

**192.168.180.2 www.ServerFreeBSD.MyCompany.ru**

1. Скорректируем файл **/etc/rc.conf**, что бы web-сервер запускался автоматически при старте системы, добавим строчку:

**apache24\_enable="YES"**

1. В файле **/etc/hosts** прописываем соответствие IP адреса и символьного имени (указываем имя хоста, которое задавали при установке)

**192.168.180.2 FreeBSDSrv.MyCompany.ru**

1. Запускаем Apache командой **apachectl start**. Если не получается и выдается сообщение об ошибки **Failed to enable the ‘httpready’ Accept Filter**, то необходимо выполнить следующие действия:

в файл **/boot/loader.conf** добавить строчку

**accf\_http\_load="YES"**

и перезагрузить систему

или

в файле **httpd.conf** добавить строчки (невозможно будет использовать фильтр)

**AcceptFilter http none**

**AcceptFilter https none**

1. Перед запуском web-сервера можно было выполнить проверку корректности конфигурационного файла с помощью команды **apachectl configtest**.
2. Проверяем, работает ли web-сервер с помощью команды **ps** или **sockstat** (вспоминаем как).
3. Переходим в каталог **/usr/local/www/apache24/data** и исправляем файл **index.html**, фразу **«It works!»** заменяем на фразу **«Ура заработало!!!»**.
4. На базовой рабочей машине открываем любой браузер, пытаемся открыть сайт по адресу [**www.serverfreebsd.mycompany.ru**](http://www.serverfreebsd.mycompany.ru). Скорее всего вы увидите абракозябры, т.к. автоматически не определилась кодировка, а у нас на сервере как помним KOI8-R. Поменяйте кодировку на странице.
5. Попробуем настроить виртуальный хостинг для сайта другой компании (имя – www.SuperFirma.ru), который так же будет расположен на нашем сервере. Нам необходимо выполнить следующие действия:
6. настроить секцию виртуального хостинга в файле **http.conf** (или в отдельном файле описания виртуальных хостов)
7. предоставить доступ с помощью протокола ftp только в каталог **/usr/local/www/SuperFirma**
8. создать учетную запись для администратора сайта (**SuperFirmaAdmin**) с минимальными правами и домашним каталогом **/usr/local/www/SuperFirma** (группа **nobody**, без командной оболочки – **nologin**, полные права в домашнем каталоге)
9. C последнего пункта и поднимаемся выше…☺ Пункты **3** и **2** сделайте самостоятельно.
10. При выполнении пункта 3 обратите внимании, что пользователи с shell nologin не могут подключаться по ftp, поэтому создаем в списке /etc/shells добавляем новый – /sbin/nologin-ftp. Делаем «мягкий» link на nologin.

**ln –s /usr/sbin/nologin /sbin/nologin-ftp**.

1. Правим **httpd.conf**

в секции **<Directory />** закомментировать строчку **Deny from all**

в конец файла добавляем:

**NameVirtualHost \***

описание основного и виртуального сайта

**<VirtualHost \*>**

**ServerName** [**www.ServerFreeBSD.MyCompany.ru**](http://www.ServerFreeBSD.MyCompany.ru)

**</VirtualHost>**

**<VirtualHost \*>**

**ServerName** [**www.SuperFirma.ru**](http://www.SuperFirma.ru)

**DocumentRoot /usr/local/www/SuperFirma/data**

**</VirtualHost>**

1. Проверяем корректность файла и перезапускаем web-сервер **apachectl graceful**.
2. Проверяем работоспособность сайтов в браузере хостовой ОС.
3. Выполните инсталляцию службы **SAMBA**.
4. Для автоматического запуска процессов в файле **/etc/rc.conf** добавляем строчки

**samba\_server\_enable=”YES”**

1. Для настройки службы SAMBA необходимо создать файл **/usr/local/etc/smb4.conf**, добавим новую секцию для общих файлов

**[global]**

**workgroup = WORKGROUP**

**server string = Samba Server**

**log file = /var/log/samba4/log.%m**

**max log size = 500**

**socket options = TCP\_NODELAY**

**os level = 1**

**map to guest = Bad Password**

**[DATA]**

**comment = Public Folder**

**path = /CommonFiles**

**public = yes**

**writable = yes**

**read only = no**

**create mask = 0777**

**directory mask = 0777**

1. Самостоятельно добавьте новую учетную запись (adduser) **CommonSamba** (оболочка – **nologin**, пустой пароль – **empty**).
2. Создайте директорию **/CommonFiles**.
3. Устанавливаем права на директорию

**сhmod –R –v 777 /CommonFiles**

1. Владельцем директории устанавливаем пользователя **CommonSamba**:

**сhown –R –v CommonSamba:CommonSamba /CommonFiles**

1. Создаем пользователя для службы SAMBA с пустым паролем

**smbpasswd -a CommonSamba**

1. Запускаем процесс SAMBA

**services samba\_server start**

1. Проверяем, запустились ли процессы связанный со службой SAMBA

**sockstat –v | grep bd**

1. На хостовой машине откройте «Мой компьютер» и в адресной строке наберите [\\192.168.180.2](file:///\\192.168.180.2) (IP Вашей виртуальной машины) и нажмите Enter. В ответ на запрос логина и пароля введите имя пользователя CommonSamba и нажмите ОК. Должны получить список доступных общих папок, попробуйте зайти в папку **common**.
2. Перейдите в виртуальную машину и выполните команду (сможете увидеть подключенных пользователей)

**smbstatus**

1. Следующая команда позволит увидеть список пользователей службы

**pdbedit –L**

1. Устанавливаем пакет для управления сервером в целом и SAMBA в частости.

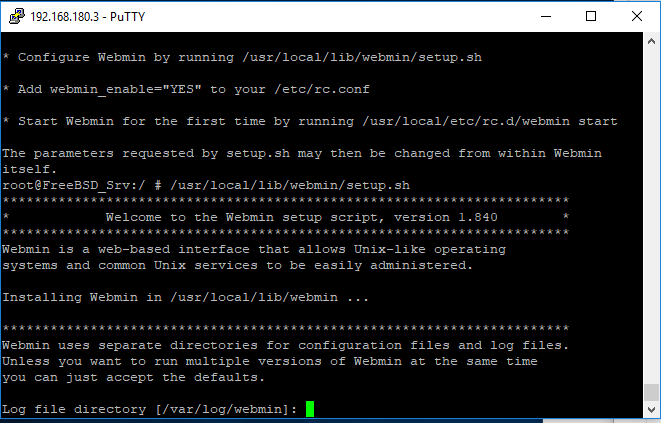
**pkg search webmin**

**pkg install webmin**

1. Добавим webmin в «автозагрузку»

**echo 'webmin\_enable="YES"' >> /etc/rc.conf**

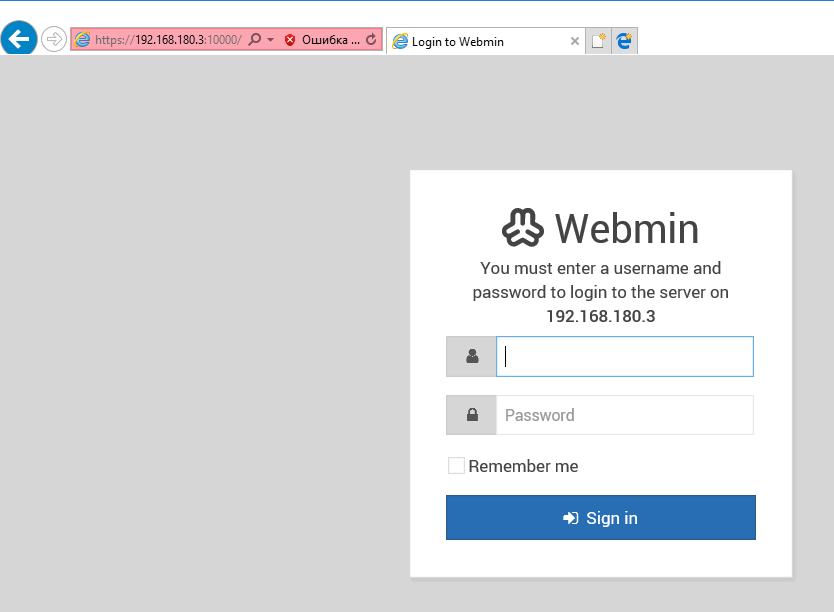
1. Запускаем файл конфигурации, в процессе ответов на вопросы не забудьте включить поддержку SSL.



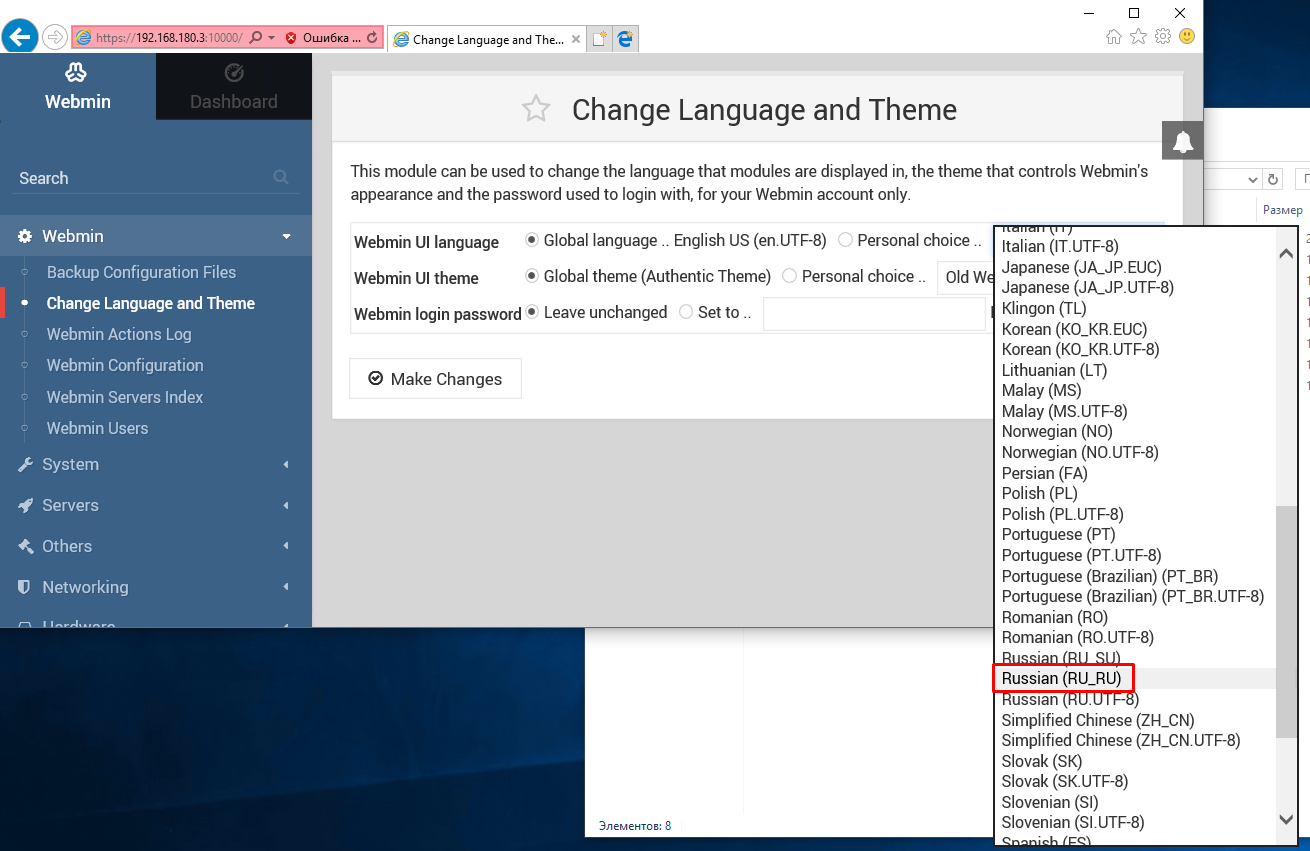
1. Запускаем webmin

/**usr/local/etc/rc.d/webmin start**

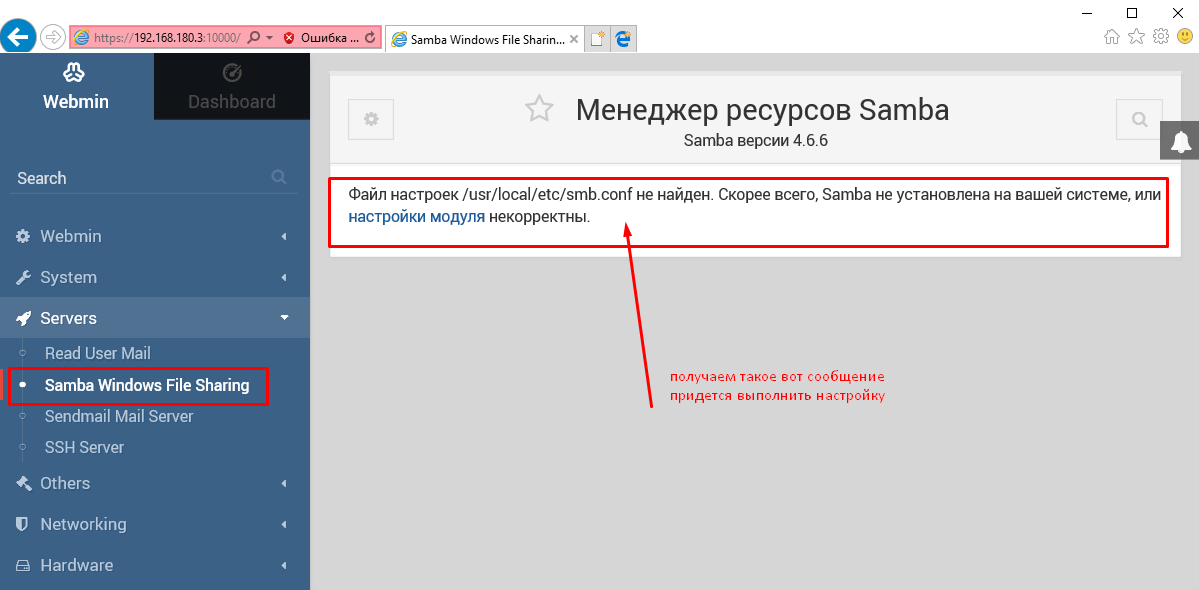
1. В браузере вводим адрес нашего сервера с указанием порта (при конфигурации порт не меняли и оставили по-умолчанию равным 10000), вводим логин и пароль (не обращаеме внимание, на замечания о неизвестном сертификате).

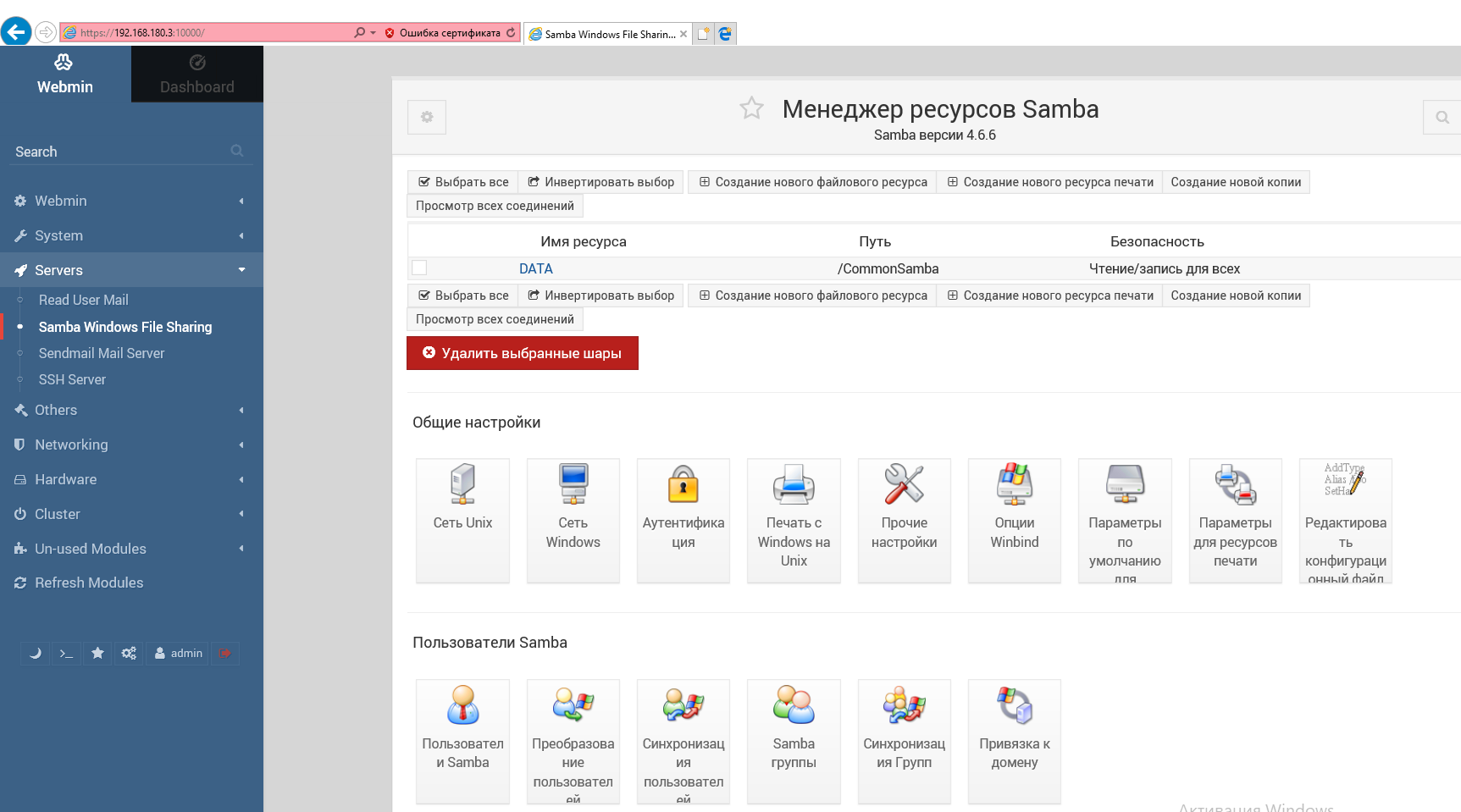


1. Для удобства работы помеянем язык консоли управления на «русский», хотя русификация частичная, поэтому польза от нее сомнительная.



1. Попытаемся управлять SAMBA через web-интерфейс. Выбираем пункт **Samba Windows File Sharing** в группе **Servers** и ничего не получается. Выполним настройку модуля – укажем корректное расположение файла конфигурации SAMBA - **/usr/local/etc/smb4.conf**.





1. Для пользователя **CommonSamba** самостоятельно настройте еще одну общую папку на сервере FreeBSD.

# Лабораторная работа 8 Работа в ОС FreeBSD. Обработка текста: grep, sed, awk.

**Цель**

Научиться применять для обработки текста инструменты FreeBSD: **grep, sed, awk**.

**Теоретические сведения**

**Утилита grep**

Утилита **grep** выполняет поиск образца в текстовых файлах и выдает все строки, содержащие этот образец.

* **-b** – Предваряет каждую строку номером блока, в котором она была найдена. Это может пригодиться при поиске блоков по контексту (блоки нумеруются с 0).
* **-c** – Выдает только количество строк, содержащих образец.
* **-h** – Предотвращает выдачу имени файла, содержащего сопоставившуюся строку, перед собственно строкой. Используется при поиске по нескольким файлам.
* **-i** – Игнорирует регистр символов при сравнениях.
* **-l** – Выдает только имена файлов, содержащих сопоставившиеся строки, по одному в строке. Если образец найден в нескольких строках файла, имя файла не повторяется.
* **-n** – Выдает перед каждой строкой ее номер в файле (строки нумеруются с 1).
* **-s** – Подавляет выдачу сообщений о не существующих или недоступных для чтения файлах.
* **-v** – Выдает все строки, за исключением содержащих образец.
* **-w** – Ищет выражение как слово, как если бы оно было окружено метасимволами \< и \>

**Примеры использования**

**Поиск всех вхождений слова.** Чтобы найти все вхождения слова "Posix" (независимо от регистра) в файле text.mm и выдать номера соответствующих строк: **/usr/bin/grep -i -n posix text.mm**

**Поиск пустых строк.** Чтобы найти все пустые строки в стандартном входном потоке: **grep ^$** или **grep -v**

Вывести из файла words.txt все строки, начинающиеся с буквы «a»:

**grep '^a' 'words.txt'**

Из результата команды [**uptime**](http://ru.wikipedia.org/wiki/Uptime) вывести только загрузку системы:

**uptime | grep -o 'load average.\*'**

Фильтр процессов без вывода команды интерпретатора (sh, csh или bash):

**ps -aef | grep -v '[с,b,a]sh'**

**Потоковый редактор sed**

**Sed** – очень легкий потоковый редактор, может выполнять операции над данными получаемыми из стандартного потока ввода.

Синтаксис команды:

**sed [-n] [-e script] [-f sfile] [files]**

Команда копирует файлы (по умолчанию со стандартного входа) на стандартный выход, редактирует их в соответствии с командами, размещенными в **script** (в командном файле или строке редактора. По флагу **-f** берет файл команд из файла **sfile**; Если есть только опция **-e script**, то флаг –e можно опустить. Флаг **–n** подавляет вывод (происходящий по умолчанию). **script** состоит из команд редактирования, по одной в строке, имеющих формат:

**[addr[, addr]] cmd [args]**

Адреса [ addr [, addr]] – это либо номера строк, либо последняя строка (символ "$"), либо регулярные выражения. Если адреса не указаны – просматриваются все входные строки. Если один адрес, то выбираются совпадающие строки. Если заданы два адреса, выбираются строки в заданном интервале.

**Символы для регулярных выражений.**

|  |  |
| --- | --- |
| **^** | Начало строки |
| **$** | Конец строки |
| **.** | Любой символ |
| **\*** | Ноль или более совпадений с символом, предваряющим \* |
| **[]** | Совпадение с любым из символов, заключенным в [ ] |
| **Примеры** | |
| **/./** | Совпадение с любой строкой, содержащей хотя бы один символ. |
| **/../** | Совпадение с любой строкой, содержащей хотя бы два символа. |
| **/^#/** | Совпадение с любой строкой, начинающейся с символа '#'. |
| **/^$/** | Совпадение с любой пустой строкой. |
| **/}$/** | Совпадение с любой строкой, которая заканчивается символом '}' (без завершающих пробелов). |
| **/} \*$/** | Совпадение с любой строкой, которая завершается символом '}' за которым следует ноль или более пробелов. |
| **/[abc]/** | Совпадение с любой строкой, которая содержит любой из следующих символов в нижнем регистре: 'a', 'b' или 'c'. |
| **/^[abc]/** | Совпадение с любой строкой, которая начинается с любого из следующих символов в нижнем регистре: 'a', 'b' или 'c'. |

**Команды sed**

Примечание: исходный файл с именем **my\_file** имеет вид:

1111

2222

3333

4444

|  |  |  |
| --- | --- | --- |
| **Команда** | **Описание** | **Пример** |
| **a\**  **text** | Добавляет text после указанной строки | **who | sed ‘2a\**  **my text**  **‘**  еще пример  **sed ‘1a\**  **my\_text**  **‘ my\_file** |
| **b label** | Осуществляет переход к команде «label:cmd» Если метка («label») отсутствует, то переход на конец командного файла. | **who | sed ‘2a\**  **my text**  **b lb1**  **2d**  **: lb1**  **3d**  **'**  еще пример  **sed ‘2a\**  **my text**  **b lb1**  **2d**  **: lb**  **3d**  **' my\_file** |
| **c\** | Замена строки или группы строк | **sed ‘2c\**  **my text**  **‘ my\_file** |
| **d** | Удалить строку, вывести следующую строку | **sed ‘2d**  **‘ my\_file** |
| **i\** | Вставка текста перед указанной строкой | **sed ‘2i\**  **my text**  **‘ my\_file** |
| **p** | Вывести строки | **sed –n ‘3p**  **‘ my\_file** |
| **q** | Прекратить обработку строк, выйти | **sed ‘**  **q**  **‘ my\_file** |
| **s/*old/new/f*** | Заменить ***old*** на ***new***.  Если ***f=g***, тогда заменить все вхождения  Если ***f=p*** – выход  Если ***f=w файл****, записать* ***файл*** | **sed 's/1111/my text/g' my\_file** |
| **w *file*** | Записать строку в файл ***file****.* | **sed '/1111/w temp' my\_file** |
| **y/*str1*/*str2*/** | Заменить **каждый** символ строки ***str1*** на соответствующий символ строки ***str2*** | **sed 'y/1234/4321/' my\_file** |
| = | Выдать текущую нумерацию входной строки | **sed '=' file** |
| **!cmd** | Выполнить команду, только если строка не выбрана | **sed ‘2,4!d’ my\_file** |

**Утилита awk**

**awk –** это:

* фильтр, который можно вызывать из командной строки так же, как и любой другой фильтр;
* программная среда, с помощью которой можно создавать новые фильтры.

Команда **awk** принимает в качестве аргументов инструкцию фильтра и список имён файлов. Инструкция заключается в одинарные кавычки и состоит из двух сегментов: **образца (условия)** и **действия**.

**Действия** заключаются в фигурные скобки.

**Синтаксис команды:**

**awk** 'образец {действие}' имена\_файлов

Утилита **awk** получает данные либо **из файлов**, либо **со стандартного устройства ввода**.

**образец** – это шаблон обрабатываемых строк.

**действие** – что будет сделано с найденной по шаблону строкой.

Если **имена\_файлов** не указаны, то входные данные берутся со стандартного устройства ввода.

Если **действие** не указано, то строка отображается на стандартном устройстве вывода (экране).

Если не задан **образец**, то действие выполняется со всеми строками.

Файл **db.dat** имеет следующее содержание:

Jim Fost student 11.10.87 5 football

Mike Brown student 12.09.86 6 football

John Rodd student 05.09.86 3 swimming

Bryan Talbot lecturer 04.07.75 10 basketball

Nick Shaw student 03.08.87 5 swimming

Ann Brook lecturer 08.04.79 6 swimming

Jane Fell student 14.05.83 2 volley-ball

BMW : Paul : Brown : 12389

BMW : Jane : Tod : 34567

Chrysler : Mike : Bell : 67890

Chevrolet : Ben : Walsh : 45634

5:30 - подъем

...

...

13:50 - обед

...

...

22:30 - отбой

**Примеры:**

**awk '/student/ {print}' db.dat**

на экран будут выведены все строки содержащие слово **paint** из файла **db.dat**

**awk '/student/' db.dat**

то же что и в предыдущим примером, но **действие** выбрано по умолчанию.

**awk '{print}' db.dat**

выводит на экран весь файл **db.dat**

**awk** может использовать в **образце** специальные символы.

**Примеры:**

**awk '/BMW/ {print}' db.dat**

ищет строку с образцом **paint**, и выводит ее на экран.

**awk '/^Mike/ {print}' db.dat**

поиск **образца** производится в **начале** каждой строки.

**awk '/ball$/ {print}' db.dat**

поиск **образца** выполняется в **конце** строки.

**Другие специальные символы:**

* **точка** – обозначает любой символ
* **звёздочка** – несколько экземпляров образца
* **квадратные скобки** – набор символов
* **"+"** – вариант "\*", обозначает один или несколько экземпляров символа
* **"?"** – вариант "\*", обозначает ни одного или один экземпляр символа
* **"|"** – позволяет задать несколько образцов

**Примеры**:

**awk '/.W/ {print}' db.dat**

ищет строку, в которой за произвольным символом следует символ **"W"**

сравните результат с предыдущим примером:

**awk '/B.W/ {print}' db.dat**

**awk '/[0-9]\*.50/ {print}' db.dat**

ищет строки, в которых встречаются слова с **50** **(13:50 или 3\_50)**

**awk '/BMW|Mike/ {print}' db.dat**

ищет строку, содержащую либо образец **BMW**, либо образец **Mike**

**Переменные, константы и операции**

Существует три типа переменных:

* переменные для обозначения полей
* специальные переменные
* пользовательские переменные

Первые два типа **awk** определяет автоматически, третий тип определяет пользователь. Можно определить арифметические и строковые константы. Арифметические константы состоят из цифр, а строковые - из любых символов, заключённых в двойные кавычки.

**Поле –** любой набор символов, ограниченный разделителем полей (по умолчнию в качестве разделителя используется пробел или символ табуляции).

**awk** нумерует поля начиная с 1. Имя переменной, обозначающей поле, состоит из знака $ и номера поля ($2 обозначает второе поле, $0 – переменная, которая содержит все поля).

**Специальные переменные.**

|  |  |
| --- | --- |
| **Переменные** | **Описание** |
| NR | номер текущей строки |
| NF | число полей в текущей строке |
| RS | разделитель строк на вводе, по умолчанию "\0" |
| FS | разделитель полей на вводе, по умолчанию пробел или табуляция |
| ORS | разделитель строк на выводе |
| RSOFS | разделитель полей на выводе |
| FSOFMT | формат вывода чисел |
| FILENAME | имя входного файла |

**Примеры:**

**awk '{print NR, $2, $4}' db.dat**

на экран будут выведены все строки, но только с полями номер 2 и 4, а перед каждой строкой будет стоять её номер.

**awk '{print $4, $3, $2, $1; print $0}' db.dat**

все строки файла будут выведены на экран два раза:

первый раз – будут выведены только поля с первого по четвертое и в обратном порядке;

второй раз – вся строка.

**awk 'BEGIN {ORS=">"} {print}' db.dat**

будут выводиться все строки файла, но после каждой строки будет ставиться символ **">"**.

**awk '{print} END {print("FileName=" FILENAME)}' db.dat**

после вывода файла будет выводиться имя файла.

**awk 'BEGIN {FS=" "} {print NF,$0}' db.dat**

входной файл имеет в качестве разделителей полей символ **"."**.

Остальные переменные используются аналогично.

**awk '{best=$2; print best}' db.dat**

переменной best присвоено значение второго поля и на экран выводится переменная best, т.е. второе поле.

**awk 'BEGIN {a=0} {print $0; a++} END {print("Всего записей:", a)}' db.dat**

будут выводиться все записи файла и одновременно подсчитываться количество записей, переменнaя **"a"**.

**awk '($1 ~ /nn/) {print}' db.dat**

в первом поле выполняется поиск образца **"nn"**

**awk '($1 !~ /BMW/) {print}' db.dat**

выбираются все записи, первое поле которых не содержит **"BMW"**

**awk '($6 ~ /[Bb]all/) {print}' db.dat**

в шестом поле ищется образец **ball** начинающийся со строчной или с прописной буквы.

**awk 'BEGIN {print "Начали..."} {print} END {print("Итого записей – "), NR}' db.dat**

Ключевое слово **BEGIN** задаёт действия, подлежащие выполнению перед обработкой строк, а **END** - действия, подлежащие выполнению после обработки строк.

**Встроенные функции**

В **awk** существует ряд встроенных функций, которые похожи на аналогичные фунции СИ.

|  |  |
| --- | --- |
| int(x) | целая часть x с усеченными лидирующими нулями |
| sqrt(x) | квадрат x |
| rand(x) | случайное число между 0 и 1 |
| srand(x) | x - новое начальное значение для rand() |
| gsub(r,s,t) | заменить строку s для каждого найденного регулярного выражения r в строке t; возвращает количество  замен; если t опущено, то используется $0 |
| index(s,t) | возвращает индекс строки t в строке s, или 0, если нет вхождений строки t |
| length(s) | возвращает длину строки s |
| match(s,r) | возвращает позицию s, в которой встретилось регулярное выражение r; возвращает 0, если r не найдено |
| split(s,a,r) | разбить строку s в массив a по регулярному выражению r; возвращает количество полей; если r опущено, то используется значение FS |
| sprints(fmt,expr-list) | печатает expr-list в соответствии с fmt, возвращает результирующую строку |
| sub(r,s,t) | аналогично gsub, за исключением того, что заменяется только первая найденная подстрока |
| substr(s,i,n) | возвращает подстроку n, начина- ющуюся с i; если n опущено, то используется остаток s |
| printf | вывод форматированного текста |
| close (filename) | закрыть файл |

**Примеры.**

**awk 'BEGIN {a=rand()} {print} END {print("Случайное число:", a)}' db.dat**

после вывода записей будет выводиться случайное число

**awk '{gsub(/BMW/,"Лада Калина");print}' db.dat**

при выводе записей слово "**BMW**" будет заменяться на "Лада Калина"

**awk '{print($0,index($0,"im"))}' db.dat**

выводятся все записи файла и за каждой записью выводится с какой позиции начинаются символы **"im"**, а если они не существуют то выводит **0**

**Файлы инструкций awk**

Так как awk является языком программирования, то инструкции можно помещать в файл, который может читать awk.

Опция **-f** позволяет awk читать инструкции из файла, а не из командной строки.

**Пример файла инструкций для awk:**

**#file: srch\_awk**

**BEGIN {print "Поиск в файле…"; count=0;} ($3 ~/stud/) {count=count+1; print;}**

**END {print "Всего найдено записей - ", count}**

После запуска командой:

**awk –f srch\_awk db.dat**

программа выводит все записи файла **db.dat**, но перед выводом записей напишет надпись " **Поиск в файле…**", а после вывода записей количество соответствующих образцу.

**Управляющие структуры**

В утилите awk есть набор управляющих структур, обеспечивающих возможность повторного выполнения команд и выбора их из нескольких вариантов.

Циклы организуются с помощью структур **while** и **for**.

Управляющая структура для выбора варианта **if**.

|  |  |
| --- | --- |
| **Управляющая структура** | **Описание** |
| if (выражение) действие  else действие | Используется для выполнения **действия**, если **выражение** - **истина**.  Если **выражение** - ложь, выполняется **действие** после **else** |
| while(выражение) действие | Цикл выполняется пока **выражение истина** |
| for(выр1;выр2;выр3) действие | Выполняется пока условие **выр2** - истина.  **выр1** выполняется перед началом цикла,  **выр3** выполняется после каждого выполнения действия |
| next | прекращает операции с текущей записью и переходит к следующей |
| exit | прекращает всю обработку и выполняет действия, указанные после END |

**Примеры:**

**awk '{i=split($0, Name, ":"); for (j=1; j<=i; j++) print ("Name[" j "]=" Name[j])}' db1.dat**

из файла, в котором резелителем полей является символ **":"**, все поля записываются в массив, который далее выводится на экран.

**awk '{ if (NF>6) print ($0); else print ("---");}' db.dat**

выводит записи в которых полей более 6-х, иначе выводит "---".

**awk '{a=1; while(a<3) {print($0);a++}}' db.dat**

выводит каждую запись по два раза.

**Создание пользовательских фильтров с помощью awk**

С помощью awk можно создавать собственные фильтры, помещая всю инструкцию в сценарий shell. Этот файл затем можно сделать исполняемым, и его имя станет новой командой. В сценарии инструкции awk следует брать в кавычки. В файле сценария инструкцию нужно записывать так, как будто вы собираетесь выполнять её из командной строки. Также можно записать образец и действия в отдельных строках. Это значит, что первая строка awk – операции будет начинаться с ключевого слова awk, а за ним в этой же строке будет стоять одинарная кавычка. Затем в отдельных строках можно указать образец и действие. После закрывающей одинарной кавычки в той же строке нужно указать имена файлов.

**Пример:**

awk '{

for(i=1;i<=3;i++)

{

printf("%s\t", $i);

}

printf("\n");

}' db.dat

Если это будет содержание файла **View\_awk**, то если его сделать запускным с помощью **chmod**, то его можно будет выполнять как команду. И эта команда будет выводить на экран из файла **db.dat** три первых поля.

**Задания по sed**

* Найти в файле **my\_file** строки, содержащие шаблон **'2222'**, и удалить их.
* Подсчитать количество строк в файле **my\_file**.
* «Раскидать» строки, содержащие шаблон **'1'** в файл с именем **1str**, шаблон **'2'** – в файл **2str**, шаблон **'3'** – в файл **3str**, шаблон **'4'** – в файл **4str**.
* Закодировать текст. Алфавит: **abcdefgh**, соответствующий код: **12345678** (т.е. **a=1, b=2,..,h=8**)
* Найти в файле **my\_file** строку по шаблону **'2222'** и заменить ее на строку **New\_Text**.

**Задания по awk**

* Напечатать последнее поле каждой строки файла.
* Вывести на экран все записи, начиная с пятой.
* Вывести на экран все записи, в которых больше 6-и полей.
* Напечатать общее число полей в файле.
* Напечатать общее число строк содержащих "**BMW**".
* Напечатать строки, первое поле которых более 5 символов и вывести перед строкой количество символов в этом поле.
* Добавить строки содержащие «**student**» в файл «**stud\_skl**».
* Вывести на экран вначале 2-ое, затем 1-ое и в конце 3-е поле.

# Лабораторная работа 10 Планирование адресного пространства в IP сетях.

**Цель**

Изучить основы эффективного планирование адресного пространства в IPv4 сетях.

**Теоретические сведения**

Типы адресов стека TCP/IP:

**- Локальные** (аппаратные, физические). Используются для адресации узлов в пределах локальной сети (подсети). Определяются сетевой технологией: Ethernet – MAC адрес интерфейса.

**- Сетевые** (IP адреса). Используются для однозначной идентификации узлов в пределах составной сети. IP адреса используются на сетевом уровне стека протоколов TCP/IP. Имеет размер 4 байта (32 бита). Состоит из двух частей:

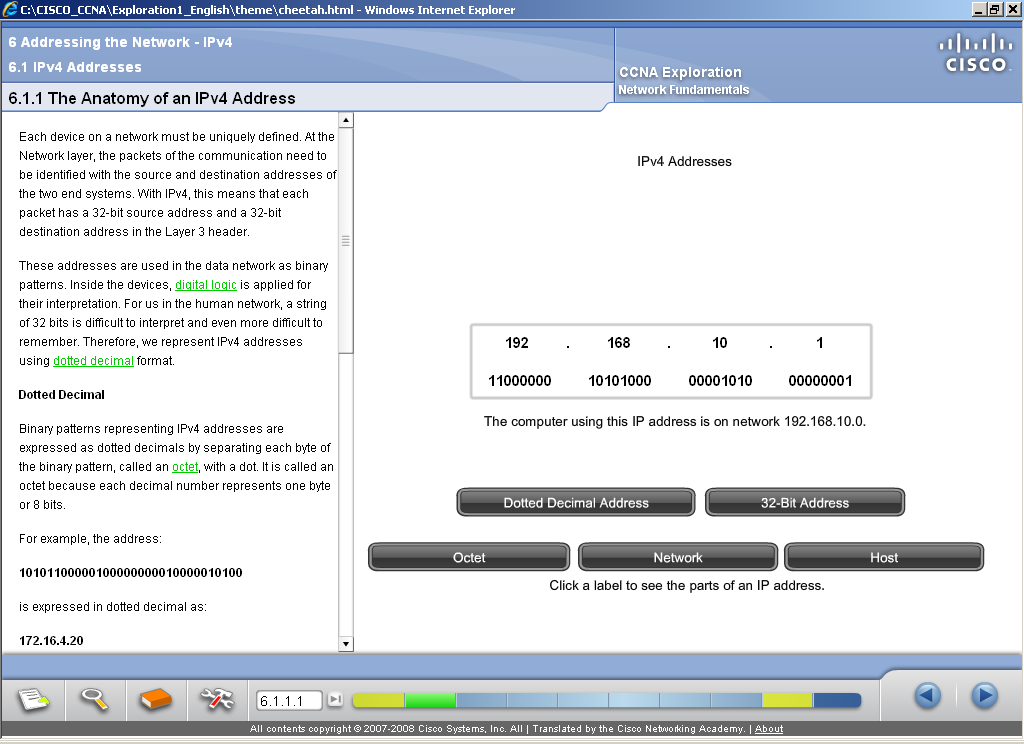
- адреса сети;

- адреса узла.

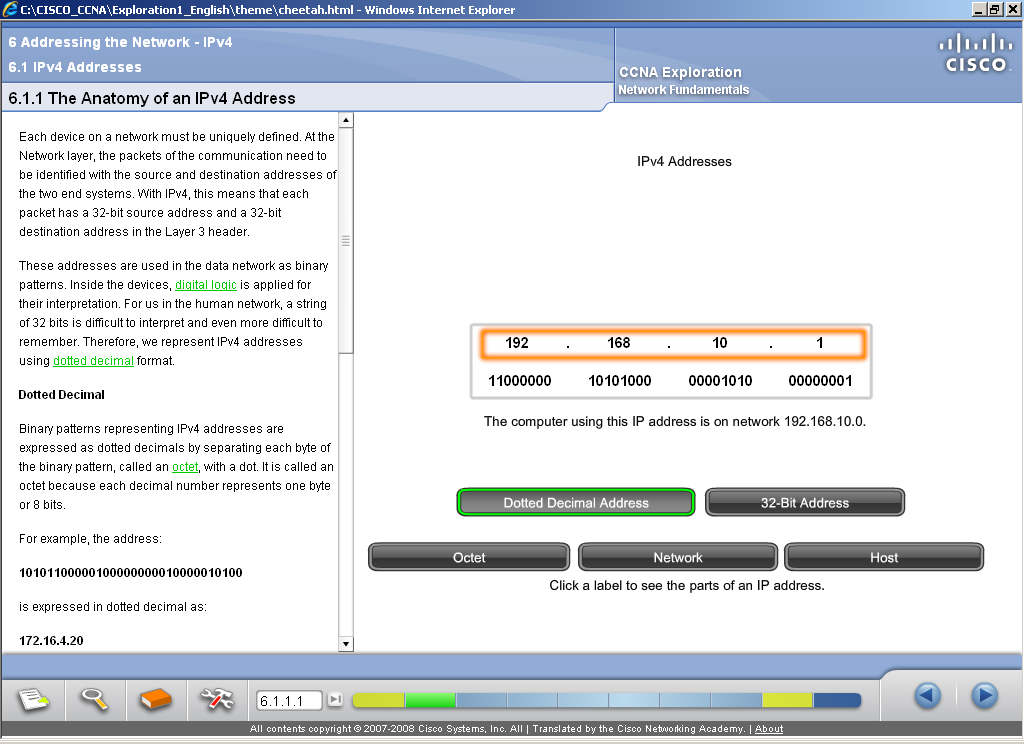
Если узел принадлежит нескольким сетям (подсетям), то он должен иметь соответствующее количество IP адресов, назначенных на интерфейсы, которые подключены к соответствующим сегментам. Т.о. IP адрес идентифицирует интерфейс сетевого устройства.

**- Символьные** (доменные, DNS адреса). Имеют иерархическую структуру. Пример, www.tstu.tver.ru. Работа системы доменных имен обеспечивается обменом информацией о соответствии символьного имени и IP адреса на основе распределенной базе данных.

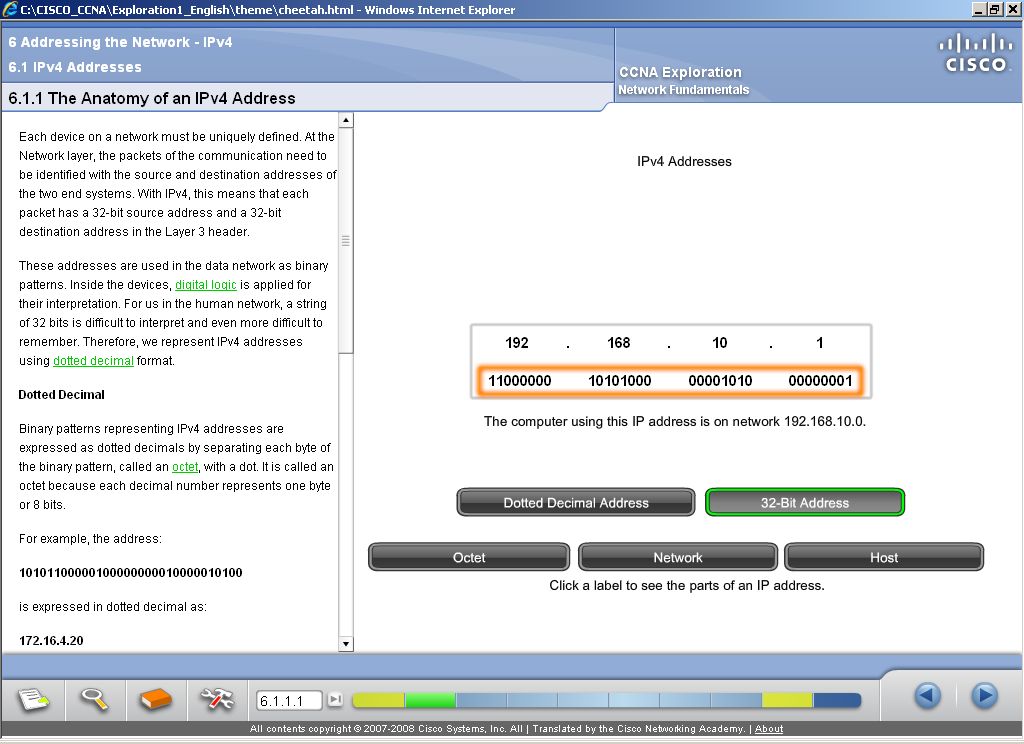
Каждое устройство в сети должно быть однозначно идентифицировано, т.е. иметь уникальный адрес. Сетевой (network) уровень стека протоколов TCP/IP использует для идентификации устройств в сети IP адреса. Соответственно адрес узла отправителя (Source Address) и адрес узла получателя (Destination Address) располагаются в заголовке IP пакета. Устройства работают с адресами в двоичном виде, используя **digital logic** для интерпретации адресов. Для человека 32-битные адреса сложны для восприятия и запоминания, поэтому используется **dotted decimal** представление адресов, т.е. представление в десятичном виде. Точка разделяет каждый байт адреса, всего получается 4 байта и соответственно 32 бита.



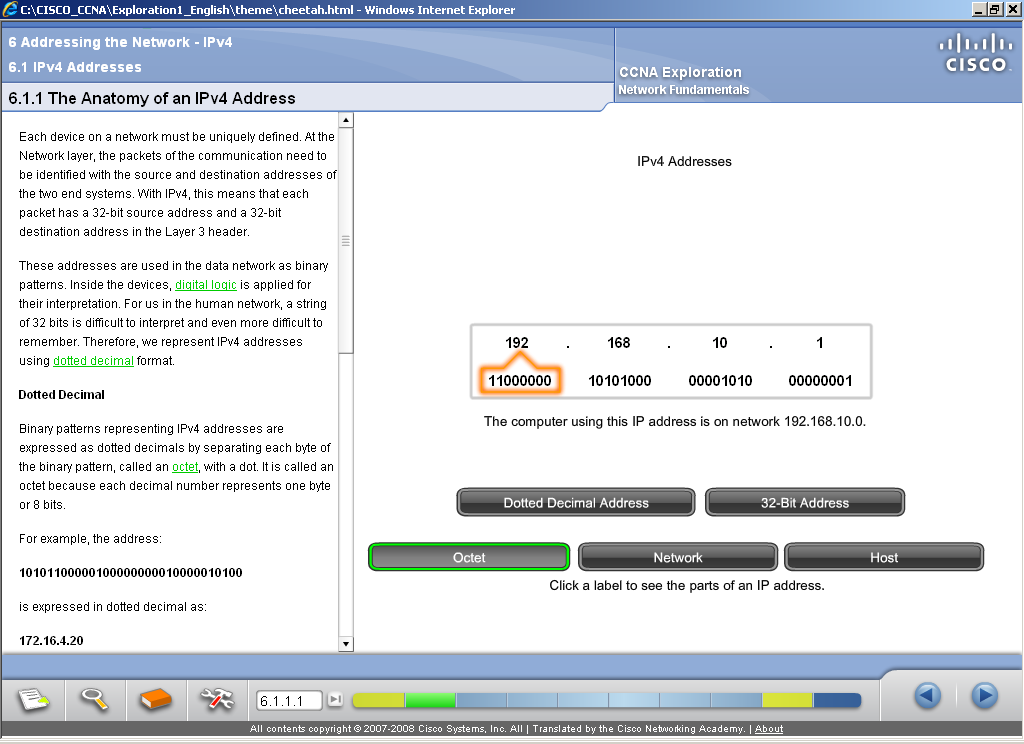
Пример представления IPv4 сетевого адреса.



Dotted Decimal адрес.

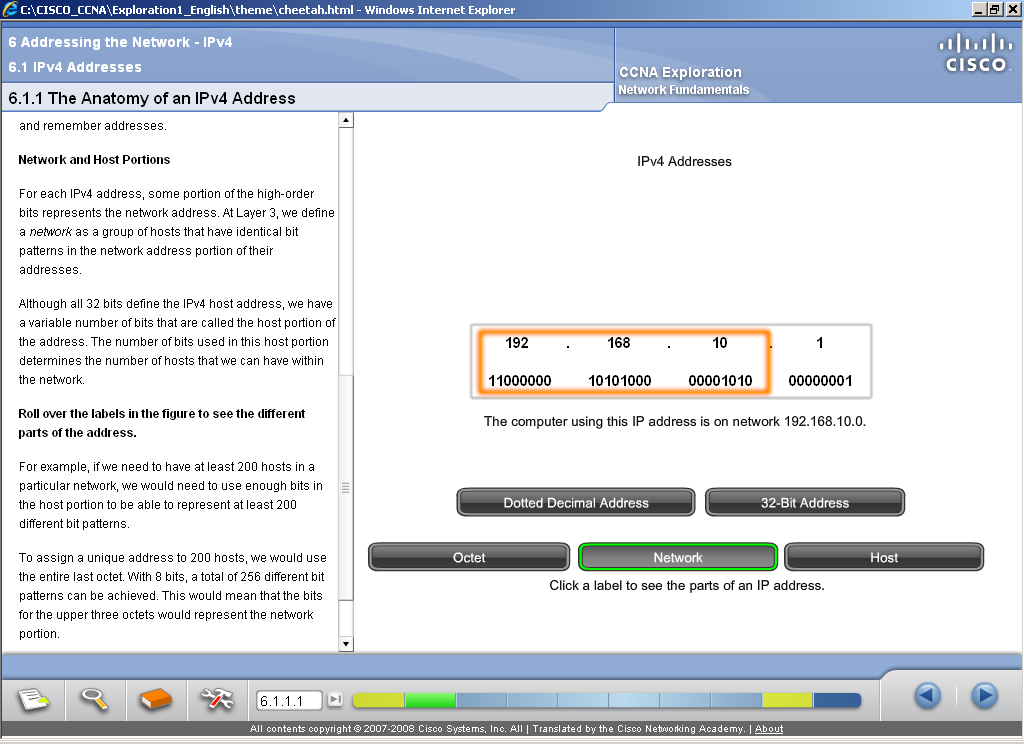


Представление в двоичном виде 32-битного адреса.

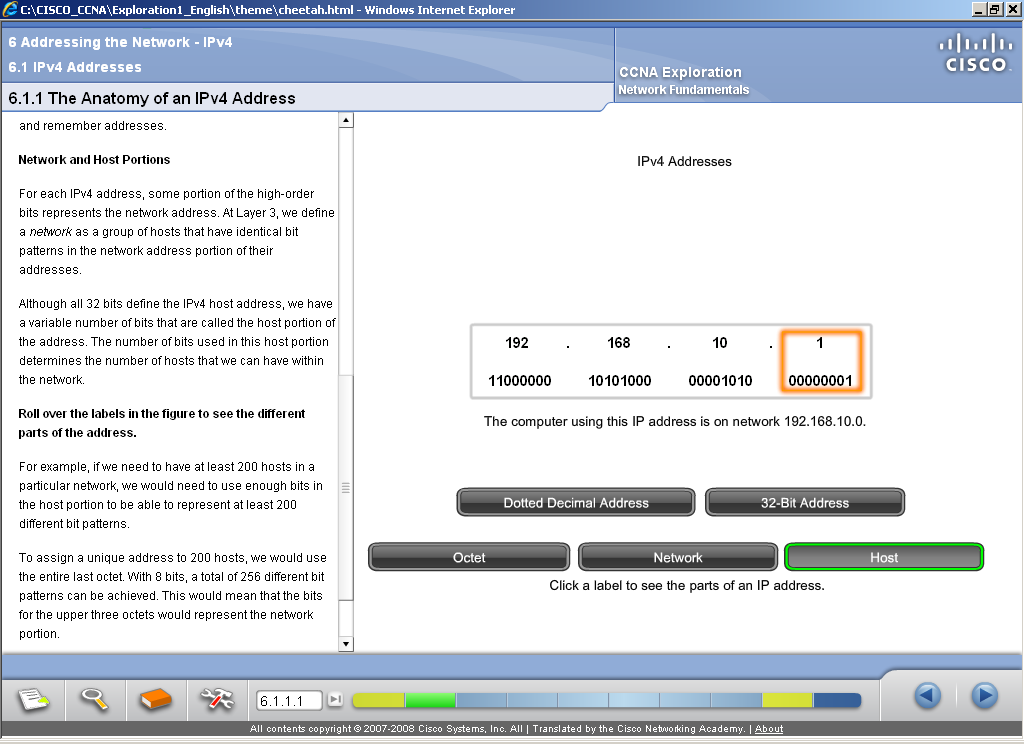


Октет (Octet).

Для каждого адреса IPv4 старшая часть битов представляет сетевой адрес (network или адрес сети). С точки зрения сетевого (network) уровня сеть определяется как группа хостов (или других сетевых устройств, работающих на этом уровне), у которых совпадает адрес сети (точнее совпадают биты в сетевой части адреса). Вторая часть адреса представляет адрес хоста в сети. Количество бит в этой части определяет максимальное количество хостов, которое может быть в пределах этой сети.



Адрес сети.



Адрес хоста.

**Перевод из двоичной системы счисления в десятичную.**

Основанием двоичной системы счисления является 2. В двоичном числе каждый бит представляет 2 в степени соответствующей порядковому номеру этого бита в числе. Например, для 8-битного числа :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |  |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Сумма =255 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 128+ | 64+ | 32+ | 16+ | 0+ | 4+ | 0+ | 1 |
| 245 | | | | | | | |

Примеры.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 59 | | | | | | | |

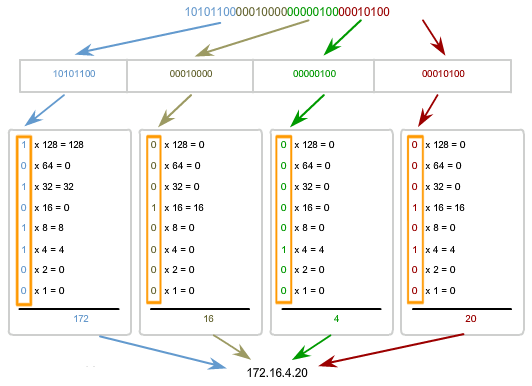
**Преобразование IPv4 адреса из двоичного представление в десятичное.**

Необходимо выполнить следующие действия:

– разделить 32-битный адрес на 4 октета по 8 бит

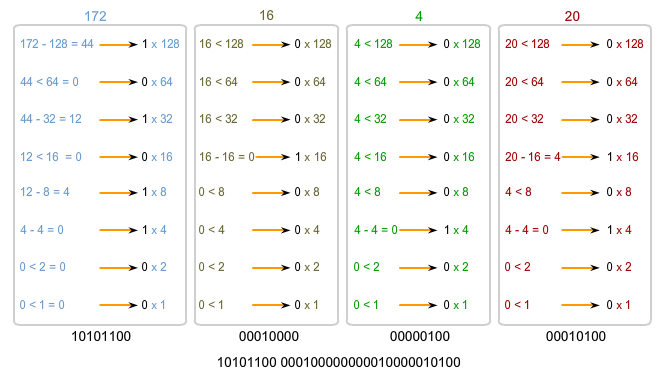
– перевести каждый октет в десятичную систему

– добавить точку между полученными десятичными числами



**Преобразование IPv4 адреса из десятичного представления в двоичное.**

Переводим каждый октет отдельно в двоичную систему счисления и соединяем полученное.



Пример.

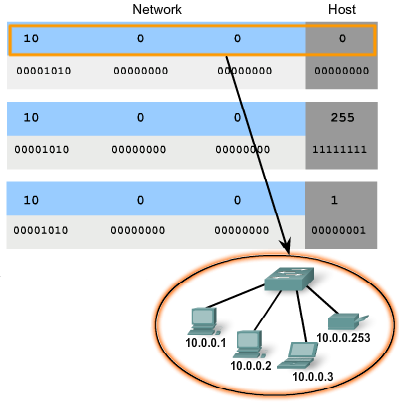
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 156 | | | | | | | |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

**IPv4 адреса для различных целей.**

В пределах интервала адресов каждой IPv4 сети выделяется три типа адресов:

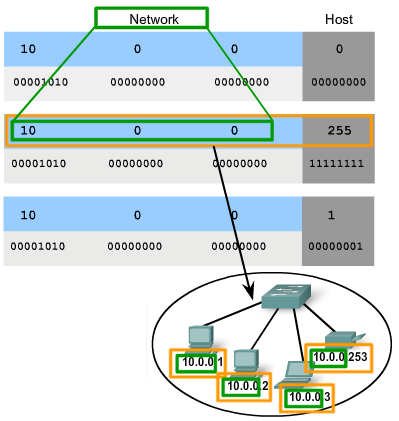
– **Network** (сетевой или адрес сети) адрес

Определяет всю сеть данного интервала, у всех хостов в данной сети будут одинаковые биты в сетевой части IPv4 адреса. В пределах интервала адресов самый меньший (все биты в части адреса хоста равны 0) зарезервирован под адрес сети.



– **Broadcast** (широковещательный) адрес

Это специальный адрес для каждой сети, который представляет все хосты в этой сети. Если какой либо хост хочет отправить пакет всем хостам сети, то в качестве адреса узла назначения он указывает этот адрес. Широковещательный адрес использует самый большой адрес в сетевом диапазоне, все биты в части адреса хоста равны 1.



– **Host** адрес

Это уникальный адрес, который идентифицирует конкретное устройство в сети, состоит из адреса сети адреса узла.

**Сетевые префиксы**

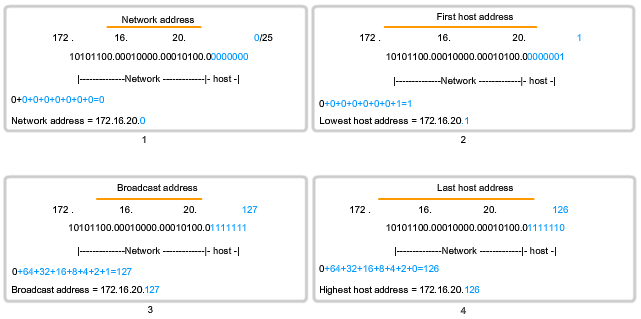
Используются, что бы указать, сколько битов в адресе будут представлять сетевую часть, а сколько хост-часть. Префиксная длина - число битов в адресе, который дает нам сетевую часть. Например, в 172.16.4.0/24, /24 - префиксная длина (первые 24 бита - сетевой адрес). При этом оставшиеся 8 бит, последний октет – это хост-часть. Кроме префиксов для определения сетевой части адреса используются маски подсети. Маска подсети состоит из 32 бит, и использует 1 и 0, чтобы указать, какие биты адреса интерпретируются, как сетевые биты и какие биты - хост биты.

Сетям не всегда назначают префикс /24. В зависимости от числа хостов в сети назначенный префикс может быть различным. Наличие различного префиксного числа изменяет диапазон адресов для хостов и широковещательный адрес для каждой сети.

|  |  |  |  |
| --- | --- | --- | --- |
| **Сеть** | **Адрес сети** | **Диапазон адресов для хостов** | **Широковещательный адрес** |
| **Количество узлов - 254** | | | |
| 172.16.4.0/24 | 172.16.4.0 | 172.16.4.1-172.16.4.254 | 172.16.4.255 |
| Двоичное представление | **10101100.00010000. 00000100.**00000000 | **10101100.00010000. 00000100.**00000001-  **10101100.00010000. 00000100.**11111110 | **10101100.00010000. 00000100.**11111111 |
| **Количество узлов – 126** | | | |
| 172.16.4.0/25 | 172.16.4.0 | 172.16.4.1- 172.16.4.126 | 172.16.4.127 |
| Двоичное представление | **10101100.00010000. 00000100.0**0000000 | **10101100.00010000. 00000100.0**0000001-  **10101100.00010000. 00000100.0**1111110 | **10101100.00010000. 00000100.0**1111111 |
| **Количество узлов – 62** | | | |
| 172.16.4.0/26 | 172.16.4.0 | 172.16.4.1- 172.16.4.62 | 172.16.4.63 |
| Двоичное представление | **10101100.00010000. 00000100.00**000000 | **10101100.00010000. 00000100.00**000001-  **10101100.00010000. 00000100.00**111110 | **10101100.00010000. 00000100.00**111111 |
| **Количество узлов – 30** | | | |
| 172.16.4.0/27 | 172.16.4.0 | 172.16.4.1- 172.16.4.30 | 172.16.4.31 |
| Двоичное представление | **10101100.00010000. 00000100.000**00000 | **10101100.00010000. 00000100.000**00001-  **10101100.00010000. 00000100.000**11110 | **10101100.00010000. 00000100.000**11111 |

**Вычисление network, broadcast и host адресов**

Рассмотрим пример вычисления сетевого, широковещательного адреса и интервал доступных для хостов адресов в сети с префиксом /25.



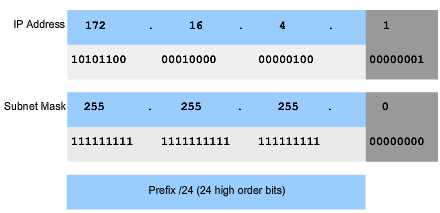
Примеры.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Последний октет в двоичном виде | Последний октет в десятичном виде | Полный адрес в десятичном виде |
| **177.111.251.218/24** | | | |
| Адрес сети | 00000000 | 0 | 177.111.251.0 |
| Широковещательный | 11111111 | 255 | 177.111.251.255 |
| Первый доступный адрес для узла | 00000001 | 1 | 177.111.251.1 |
| Последний доступный адрес для узла | 11111110 | 254 | 177.111.251.254 |
| **184.59.126.166/29** | | | |
| Адрес сети | 10100000 | 160 | 184.59.126.160 |
| Широковещательный | 10100111 | 167 | 184.59.126.167 |
| Первый доступный адрес для узла | 10100001 | 161 | 184.59.126.161 |
| Последний доступный адрес для узла | 10100110 | 166 | 184.59.126.166 |

**Использование масок и подсетей (subnet mask)**

Маска подсети используется для определения бит, которые будут представлять сетевую часть адреса и хост часть адреса.

Например, на рисунке представлен IPv4 адрес с префиксом /24 и маской подсети 255.255.255.0.



Возможные варианты маски подсети.

00000000 = 0

10000000 = 128

11000000 = 192

11100000 = 224

11110000 = 240

11111000 = 248

11111100 = 252

11111110 = 254

11111111 = 255

Для получения адреса сети по IP адресу узла и subnet mask необходимо перевести IP адрес узла и subnet mask в двоичный вид и выполнить по-битовую операцию логического умножения (AND).

Например, запись адреса хоста 172.16.4.35/27 означает следующее:

Адрес (соответственно десятичное и двоичное представление)

172.16.20.35

10101100.00010000.00010100.00100011

Subnet mask (маска подсети)

255.255.255.224

11111111.11111111.11111111.11100000

Адрес сети

172.16.20.32

10101100.00010000.00010100.00100000

Пример определения адреса сети по IPv4 адресу узла и subnet mask.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **172.16.132.70/20** | | | | |
| **Адрес узла** | **172** | **16** | **132** | **70** |
| *Двоичное представление адреса узла* | *10101100* | *00010000* | *10000100* | *01000110* |
| *Двоичное представление subnet mask* | *11111111* | *11111111* | *11110000* | *00000000* |
| *Двоичное представление адреса сети* | *10101100* | *00010000* | *10000000* | *00000000* |
| **Адрес сети** | **172** | **16** | **128** | **0** |

**Примеры определения сетевого адреса**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес узла** | **10** | **122** | **149** | **190** |
| **Subnet mask** | **255** | **255** | **255** | **224** |
| *Двоичное представление адреса узла* | *00001010* | *01111010* | *10010101* | *10111110* |
| *Двоичное представление subnet mask* | *11111111* | *11111111* | *11111111* | *11100000* |
| *Двоичное представление адреса сети* | *00001010* | *01111010* | *10010101* | *10100000* |
| **Адрес сети** | **10** | **122** | **149** | **160** |

Планирование и документирование адресного пространства в сети позволяет достигнуть следующих целей:

– предотвращение одинаковых адресов у разных устройств;

– обеспечение и контроль доступа;

– мониторинг безопасности и производительности.

При распределении адресного пространства, прежде всего, необходимо определить будут ли использоваться public адреса или только private. Как правило, провайдер (ISP) выделяет небольшой диапазон public адресов организации при подключении ее к глобальной сети (возможно, только один), поэтому все устройства локальной сети организации не могут иметь public адреса, но в этом и нет необходимости. Для обеспечения узлов локальной сети доступом в Интернет может использоваться механизм proxy-серверов и шлюзов. Если необходимо обеспечить доступ к узлам, имеющим private адреса (например, отдельным серверам), из глобальной сети, то можно использовать механизм NAT (Network Address Translation).

При распределении адресного пространства также необходимо определить общее количество узлов в сети, которым потребуются IP адреса, и в каком количестве.

Можно выделить следующие группы устройств сети, которым необходимы IP адреса:

- конечные устройства (компьютеры пользователей, компьютеры администраторов, сервера, сетевые принтеры, IP телефоны и IP камеры)

- сетевые устройства (LAN и WAN интерфейсы маршрутизаторов)

- сетевые устройства, которым необходимы IP адреса для управления ими (коммутаторы, точки доступа).

**Основы выделения подсетей**

Подсети позволяет создать несколько логических сетей из одного блока выделенных адресов.

Можно выделить несколько причин разделение сети на подсети:

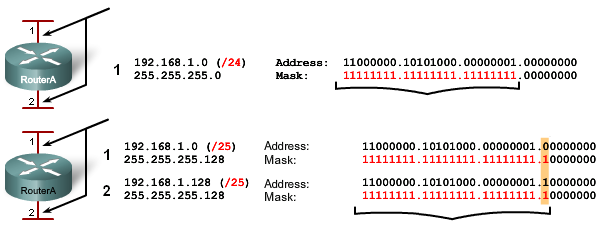
**Ограничение широковещательного трафика** – широковещательный трафик распространяется только в пределах подсети; выделяем из большой области несколько отдельных областей, тем самым сокращаем объем широковещательного трафика.

**Разделение сети в соответствии с требованиями** – различные группы пользователей объединяются в отдельные подсети в соответствии с их требованиями к сети.

**Безопасность** – выделение отдельных подсетей позволяет более гибко настраивать и управлять политикой безопасности и доступом.

При использовании маршрутизатора, у каждого его интерфейса должен быть уникальных адрес из интервала адресов данной сети. При использовании каждого дополнительного бита в адресе для идентификации сети количество доступных сетей удваивается, но при этом сокращается количество хостов, которое могут быть подключены к этой сети.

У RouterA на рисунке есть два интерфейса, чтобы связать две сети. Имея блок адреса 192.168.1.0/24, выделим две подсети. Заимствуем один бит от хост части и будем использовать маску подсети 255.255.255.128, вместо исходной 255.255.255.0.



|  |  |  |  |
| --- | --- | --- | --- |
| Подсеть | Адрес сети | Диапазон адресов для узлов | Широковещательный (broadcast) адрес |
| 0 | 192.168.1.0/25 | 192.168.1.1-192.168.1.126 | 192.168.1.127 |
| 1 | 192.168.1.128/25 | 192.168.1.129-192.168.1.254 | 192.168.1.255 |

**Расчет количества подсетей и хостов в сети**

Формула расчета количества подсетей:

2^n, где n – число бит, которое «заимствовано»

Пример: 2^1 = 2 подсети.

Формула расчета количество узлов:

2^n – 2, где n – число бит в хост части адреса

Пример: (2^7 – 2) = 126 в каждой подсети имеем по 126 хостов.

**Примеры расчета количества хостов в сети**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес сети** | **10** | **0** | **0** | **0** |
| **Subnet mask** | **255** | **255** | **255** | **240** |
| *Адрес сети в двоичном виде* | *00001010* | *00000000* | *00000000* | *0000****0000*** |
| *Subnet mask в двоичном виде* | *11111111* | *11111111* | *11111111* | *1111****0000*** |
| **Количество хостов** | **2^4-2 = 14** | | | |

**Пример расчета доступных IP адресов для узлов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес сети в десятичном виде** | **10** | **75** | **22** | **0** |
| **Subnet mask в десятичном виде** | **255** | **255** | **255** | **0** |
| *Адрес сети в двоичном виде* | *00001010* | *01001011* | *00010110* | *00000000* |
| *Subnet mask в двоичном виде* | *11111111* | *11111111* | *11111111* | *00000000* |
| Первый доступный IP адрес | 10 | 75 | 22 | 1 |
| Последний доступный IP адрес | 10 | 75 | 22 | 254 |
| Широковещательный (broadcast) адрес | 10 | 75 | 22 | 255 |
| Следующий доступный адрес сети | 10 | 75 | 23 | 0 |

**Пример с выделение 3-х подсетей**

Имеем диапазон доступных для распределения адресов - 192.168.1.0/24, необходимо выделить три подсети. Если «заимствовать» 1 бит из host-части адреса, то получим только две подсети, поэтому «заимствуем» 2 бита и используем соответственно маску подсети 255.255.255.192. По формуле расчета количества подсетей получаем 2^2 = 4, т.е. в данном случае доступно 4 подсети.

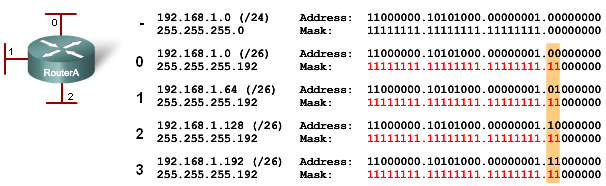
Subnet 0: 0 = **00**000000

Subnet 1: 64 = **01**000000

Subnet 2: 128 = **10**000000

Subnet 3: 192 = **11**000000

Для хост части осталось 6 бит, поэтому в каждой подсети может быть максимально - (2^6-2) = 62 хоста.



|  |  |  |  |
| --- | --- | --- | --- |
| Подсеть | Адрес сети | Диапазон адресов для узлов | Широковещательный (broadcast) адрес |
| 0 | 192.168.1.0/26 | 192.168.1.1-192.168.1.62 | 192.168.1.63 |
| 1 | 192.168.1.64/26 | 192.168.1.65-192.168.1.126 | 192.168.1.127 |
| 2 | 192.168.1.128/26 | 192.168.1.129-192.168.1.190 | 192.168.1.191 |
| 3 | 192.168.1.192/26 | 192.168.1.193-192.168.1.254 | 192.168.1.255 |

**Пример выделения 6-ти подсетей**

Имеем диапазон доступных для распределения адресов - 192.168.1.0/24, необходимо выделить три подсети. Если «заимствовать» 2 бита host-части адреса, то получим только четыре подсети, поэтому «заимствуем» 3 бита и используем соответственно маску подсети 255.255.255.224. По формуле расчета количества подсетей получаем 2^3 = 8, т.е. в данном случае доступно 8 подсетей.

0 = **000**00000

32 = **001**00000

64 = **010**00000

96 = **011**00000

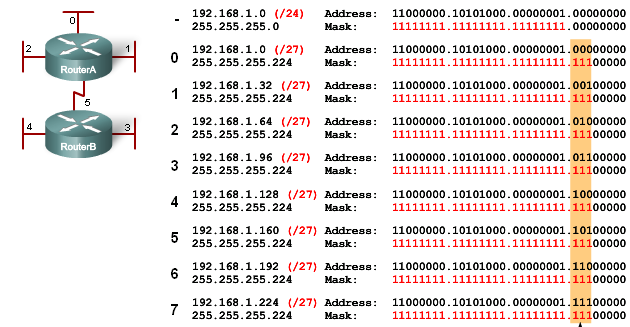
128 = **100**00000

160 = **101**00000

192 = **110**00000

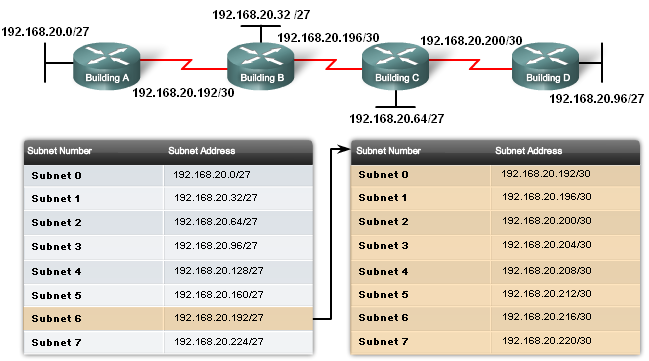
224 = **111**00000

Для хост-части осталось 6 бит, поэтому в каждой подсети может быть максимально (2^5-2) = 30 узлов.



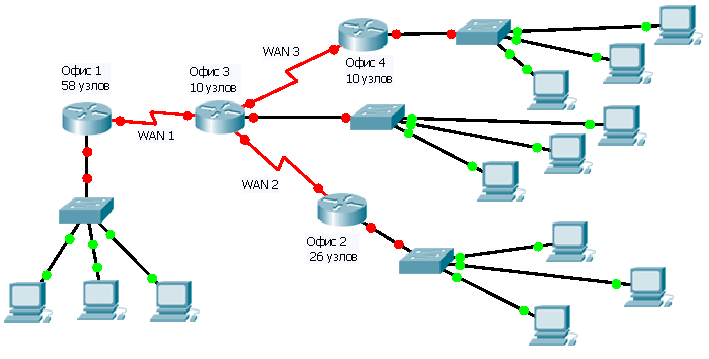
**Выделение подсетей из подсети**

Механизм выделения подсетей из подсетей позволяет более эффективно использовать пространство адресов. Для WAN подключений требуется по два адреса и нецелесообразно использовать подсеть с большим числом доступных адресов.



**Пример распределения адресного пространства № 1**

Рассмотрим следующую топологию сети организации.



Доступный диапазон адресов для распределения - **192.168.15.0/24**.

Для каждого WAN соединения используется два IP адреса.

1. Рассчитаем общее количество узлов в сети:

58+26+10+10+2+2+2 = 110

2. Имеющийся диапазон адресов позволяет адресовать 2^8-2 = 254 узла. (Для host-части доступно 32-24 = 8 бит). Соответственно имеющегося диапазона достаточно для адресации всех узлов в сети.

3. Максимальное количество хостов в подсети = 58, соответственно необходимо использовать префикс /26 – (2^6-2 = 62) при 6 битах для хост части адреса. Получаем маску подсети при 2-х «заимствованных» битах – 255.255.255.192. В этом случае можно получить только 4 подсети, которых будет недостаточно для адресации WAN соединений. Таким образом, получили ситуацию, когда при использовании одинаковой маски подсети количество подсетей недостаточно.

Более того, в этом случае получаем очень неэффективное использование адресов в подсетях.

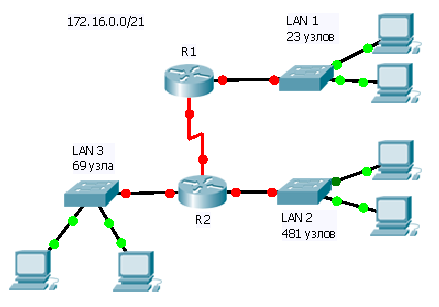
|  |  |  |
| --- | --- | --- |
|  | Используется адресов | Не используется адресов |
| Офис 1 | 58 | 4 |
| Офис 2 | 26 | 36 |
| Офис 3 | 10 | 52 |
| Офис 4 | 10 | 52 |
| WAN | 2 | 60 |

Возможным решение в данной ситуации будет использование для каждой подсети своей сетевой маски, которая будет соответствовать потребностям этой сети в IP адресах. В случае использования VLSM (маски подсети переменной длинны) получаем более «красивое» распределение адресного пространства.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Адрес сети (подсети)** | **Диапазон адресов** | **Broadcast адрес** |  |
| Офис 1 – 58 | 192.168.15.0 | .1-.62 | .63 | 192.168.15.0/26 |
| Офис 2 – 26 | 192.168.15.64 | .65-.94 | .95 | 192.168.15.64/27 |
| Офис 3 – 10 | 192.168.15.96 | .97-.110 | .111 | 192.168.15.96/28 |
| Офис 4 – 10 | 192.168.15.112 | .113-.126 | .127 | 192.168.15.112/28 |
| WAN 1 – 2 | 192.168.15.128 | .129-.130 | .131 | 192.168.15.128/30 |
| WAN 2 – 2 | 192.168.15.132 | .133-.134 | .135 | 192.168.15.132/30 |
| WAN 3 – 2 | 192.168.15.136 | .137-.138 | .139 | 192.168.15.136/30 |

**Пример распределения адресного пространства № 2**

Рассмотрим следующую топологию сети организации.



Доступный диапазон адресов для распределения - **172.16.0.0/21**.

Для WAN соединения используется два IP адреса.

При распределении будем использовать длину маски подсети переменной длины (VLSM).

Начинаем с самой большой подсети (по количеству узлов) – это LAN 2 (481 узел). Минимально достаточное число бит для адресации 481 узла равно 9 (2^9-2 = 510). При этом для сетевой части остается 32-9=23 бита, таким образом имеем префикс /23 (или subnet mask 255.255.254.0). Аналогичным образом распределяем адреса для остальных подсетей. Результаты представлены в таблице.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Адрес подсети** | **Диапазон IP адресов узлов** | | **Broadcast** |
| LAN 2 (481 узел) | 172.16.0.0/23 | 172.16.0.1 | 172.16.1.254 | 172.16.1.255 |
| LAN 3 (69 узлов) | 172.16.2.0/25 | 172.16.2.1 | 172.16.2.126 | 172.16.2.127 |
| LAN 1 (23 узла) | 172.16.2.128/27 | 172.16.2.129 | 172.16.2.158 | 172.16.2.159 |
| WAN (2 узла) | 172.16.2.160/30 | 172.16.2.164 | 172.16.2.162 | 172.16.2.163 |

**Задание**

1. Переведите число **161** из десятичной системы счисления в двоичную.

2. Перевести следующие числа из двоичной системы счисления в десятичную.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| ? | | | | | | | |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| ? | | | | | | | |

3. Определить сетевой, широковещательный и диапазон адресов для устройств по IP-адресу **152.138.77.82/28**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Последний октет в двоичном виде | Последний октет в десятичном виде | Полный адрес в десятичном виде |
| **152.138.77.82/28** | | | |
| Адрес сети |  |  |  |
| Широковещательный |  |  |  |
| Первый доступный адрес для узла |  |  |  |
| Последний доступный адрес для узла |  |  |  |

4. Определить адрес сети по IPv4 адресу узла и subnet mask.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес узла** | **10** | **124** | **132** | **71** |
| **Subnet mask** | **255** | **255** | **128** | **0** |
| Двоичное представление адреса узла |  |  |  |  |
| Двоичное представление subnet mask |  |  |  |  |
| Двоичное представление адреса сети |  |  |  |  |
| **Адрес сети** |  |  |  |  |

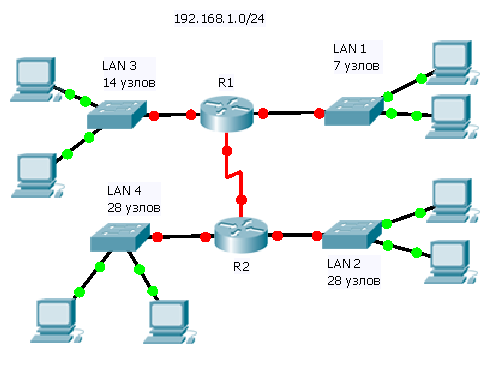
5. Рассчитать количества узлов в сети

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес сети** | **10** | **0** | **0** | **0** |
| **Subnet mask** | **255** | **255** | **240** | **0** |
| Адрес сети в двоичном виде |  |  |  |  |
| Subnet mask в двоичном виде |  |  |  |  |
| **Количество хостов** |  | | | |

6. Рассчитать диапазон IP-адресов для сети.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес сети в десятичном виде** | **10** | **75** | **128** | **0** |
| **Subnet mask в десятичном виде** | **255** | **255** | **192** | **0** |
| Адрес сети в двоичном виде |  |  |  |  |
| Subnet mask в двоичном виде |  |  |  |  |
| Первый доступный IP адрес |  |  |  |  |
| Последний доступный IP адрес |  |  |  |  |
| Широковещательный (broadcast) адрес |  |  |  |  |
| Следующий доступный адрес сети |  |  |  |  |

6. Выполнить эффективное распределение адресного пространства.



Доступный диапазон адресов для распределения - **192.168.1.0/24**.

Для WAN соединения используется два IP адреса.

При распределении будем использовать длину маски подсети переменной длины (VLSM).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Адрес подсети** | **Диапазон IP адресов узлов** | | **Broadcast** |
| LAN 2 (28 узлов) |  |  |  |  |
| LAN 4 (28 узлов) |  |  |  |  |
| LAN 3 (14 узлов) |  |  |  |  |
| LAN 1 (7 узлов) |  |  |  |  |
| WAN (2 узла) |  |  |  |  |

# Лабораторная работа 11 Знакомство с программами Boson Network Designer и NetSim.

**Замечание (важное ☺):** Для выполнения следующих ЛР можно использовать (рекомендуется) в качестве эмулятора работы сети и сетевых устройств программу PacketTracer от компании Cisco (<https://www.netacad.com/ru/courses/packet-tracer>). Для его использования зарегистрируйтесь на сайте сетевой академии Cisco и пройдите вводный курс по работе в PacketTracer (<https://www.netacad.com/ru/courses/packet-tracer/introduction-packet-tracer>). Там же Вы сможете скачать последнюю версию программы.

**Цель**

Освоить основные возможности программы Boson Network Designer, научиться, с его помощью, проектировать топологию сети.

Освоить основные возможности программного продукта Boson NetSim. В процессе выполнения лабораторной работы необходимо изучить основные возможности программы и научиться пользоваться ими.

**Теоретические сведения**

«Boson NetSim» являет эмулятором сетевых устройств компании Cisco на основе Cisco IOS. Данная программа помогает получить практические знания по работе с сетевыми устройствами: концентраторами, коммутаторами, маршрутизаторами. В состав программного продукта включена утилита для моделирования сети (построения топологии) – «Boson Network Designer».

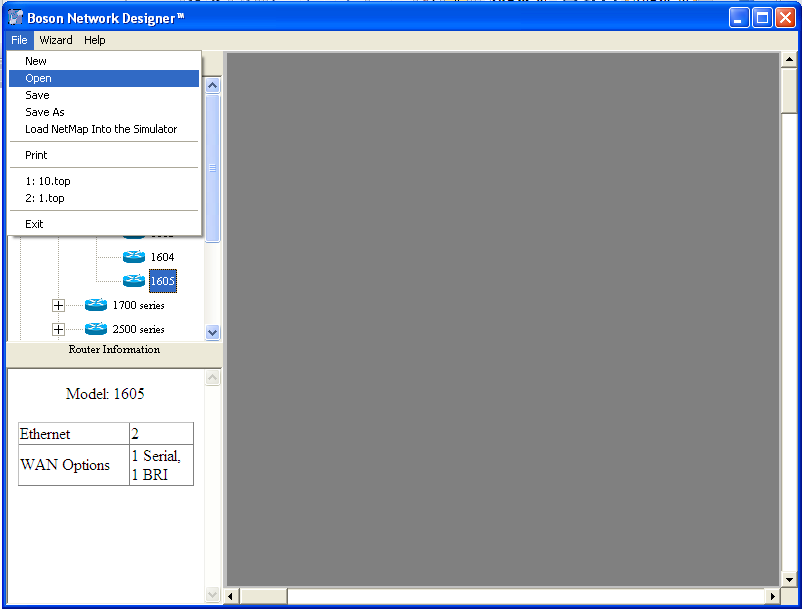


Рис. 11.1 Основное окно программы Boson Designer.

Для создания новой топологии сети можно использовать Wizard. Пункт меню «Wizard» имеет два подпункта «Make Connection Wizard» - для создания подключений (связи) между устройствами и «Add Device Wizard» для добавления нового устройства в топологию (см. Рис. 11.2).

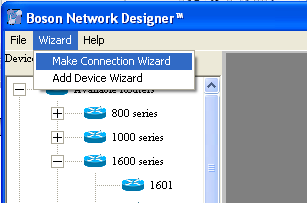


Рис. 11.2 Пункт меню «Wizard».

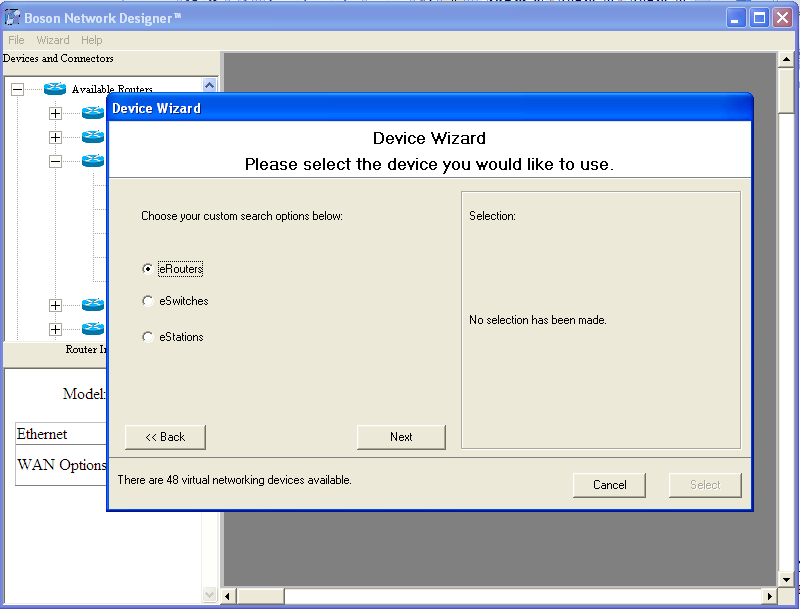
C помощью пункта «Add Device Wizard» можно добавить новое устройство в топологию и задать его параметры, последовательно отвечая на вопросы «мастера» (см. Рис. 11.3):

– тип (маршрутизатор, коммутатор, рабочая станция);

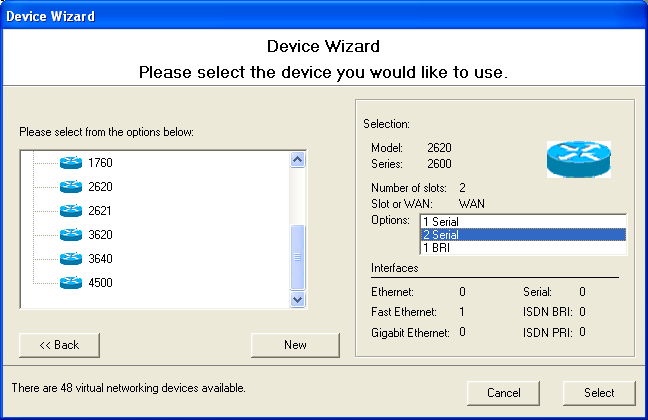
– серию;

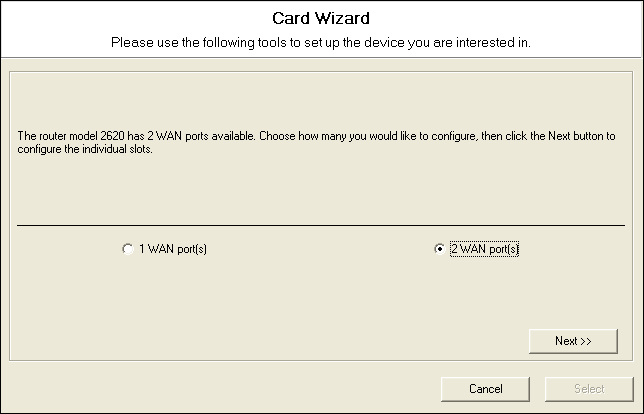
– модель;

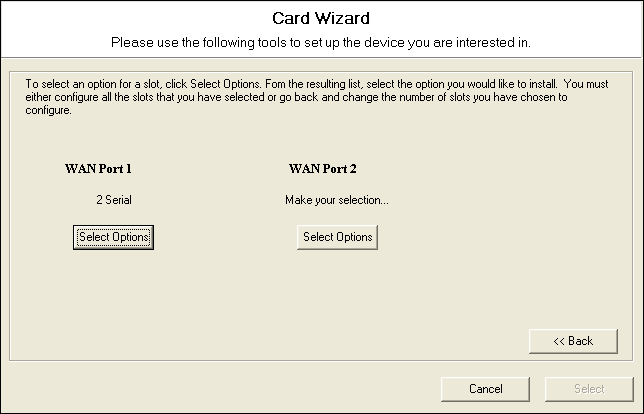
– имя.











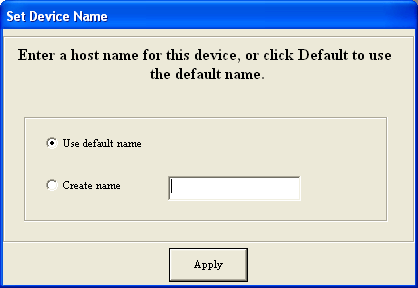


Рис. 11.3 Пошаговая настройка устройства.

Для маршрутизатора можно выбрать дополнительные параметры WAN (например, если допускается использование нескольких возможных модулей расширения).

Добавить новой устройство в проектируемую топологию можно просто «перетащив» его из левой части (список доступных устройств) на рабочую область.

С помощью пункта «Make Connection Wizard» устанавливаем различного вида соединения между устройствами в топологии. На первом шаге выбирается тип соединения (см. Рис. 11.5).

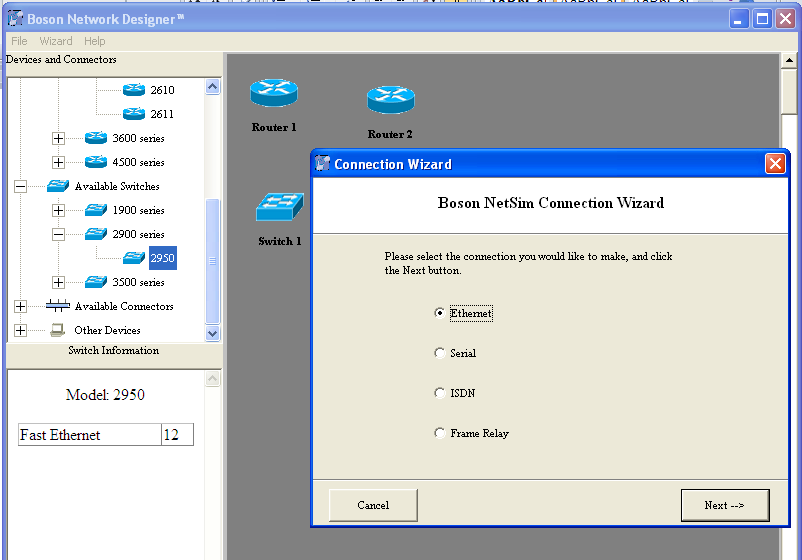


Рис. 11.5 Использование Connection Wizard. Шаг первый: выбор типа соединения.

На втором шаге выбираем первое устройство и его интерфейс (см. Рис. 11.6).

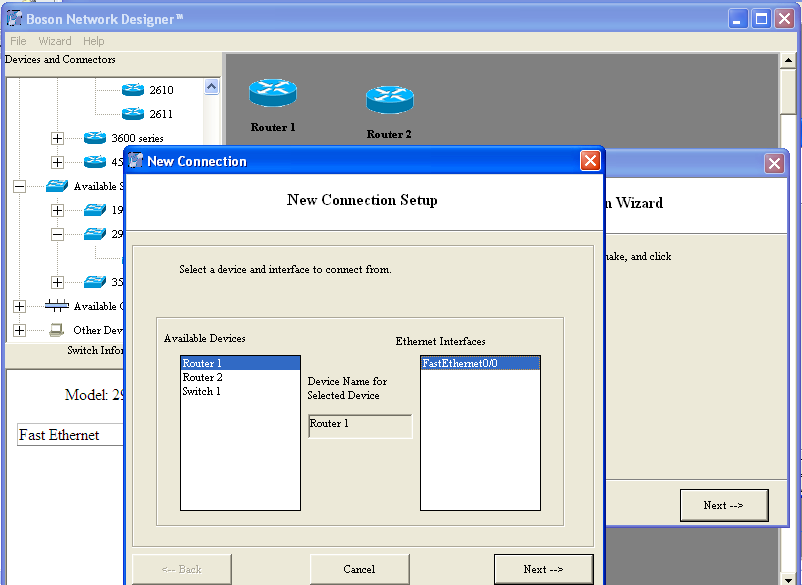


Рис. 11.6 Использование Connection Wizard. Шаг второй: выбор устройства и интерфейса.

На следующем шаге выбираем второе устройство и его интерфейс (см. Рис. 11.7).

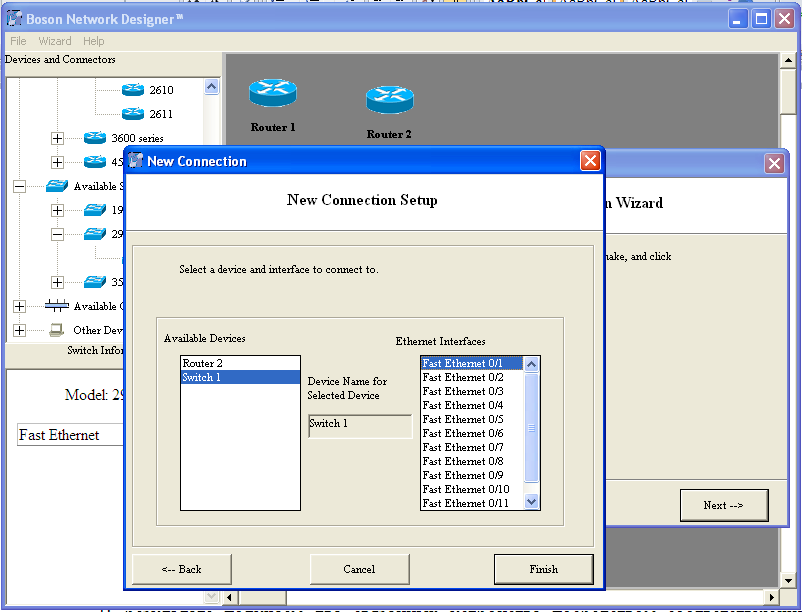


Рис. 11.7 Использование Connection Wizard. Шаг третий: выбор второго устройства и его интерфейса.

В результате получаем два связанных устройства посредством соответствующих интерфейсов. Аналогичным образом к одному устройству можно подключить несколько при условии наличия необходимых интерфейсов (см. Рис. 11.8).

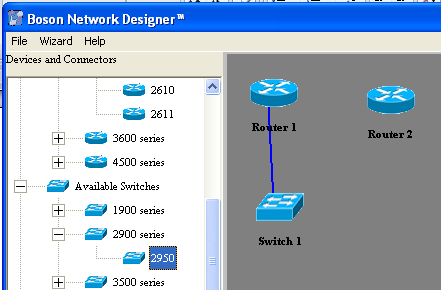


Рис. 11.8 Результат использования «Connection Wizard».

Для добавления/удаления соединения, а также удаления устройства из топологии можно пользоваться контекстным меню (клик правой кнопкой мышки на устройстве) (см. Рис. 11.9).

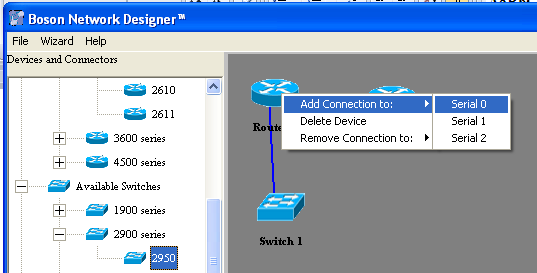


Рис. 11.9 Контекстное меню устройства.

Используя соответствующие пункты меню «File» можно сохранить топологию, загрузить ранее сохраненную топологию и загрузить созданную топологию в симулятор «Boson NetSim» (пункт меню «Load NetMap into the Simulator»).

«Boson NetSim» являет эмулятором сетевых устройств компании Cisco на основе Cisco IOS. Данная программа помогает получить практические знания по работе с сетевыми устройствами: концентраторами, коммутаторами, маршрутизаторами.

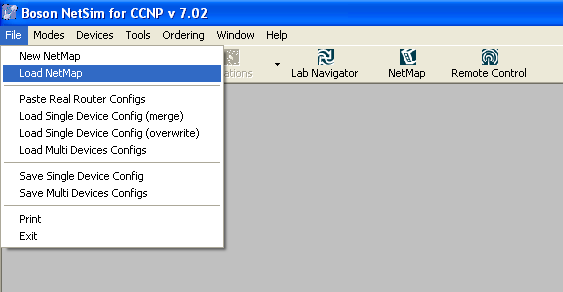


Рис. 11.10 Основное окно программы Boson NetSim.

Работа с программой начинается с создания новой (меню File/New NetMap) или загрузки существующей топологии (меню File/Load NetMap). При создании новой топологии запускается программа Boson Network Designer, которая была рассмотрена в предыдущей лабораторной работе.

После загрузки топологии сети с симулятор пользователь получает возможность производить настройку сетевых устройств, спроектированной сети в режиме интерфейса командной строки (Command Line Interface - CLI) (см. Рис. 11.11).

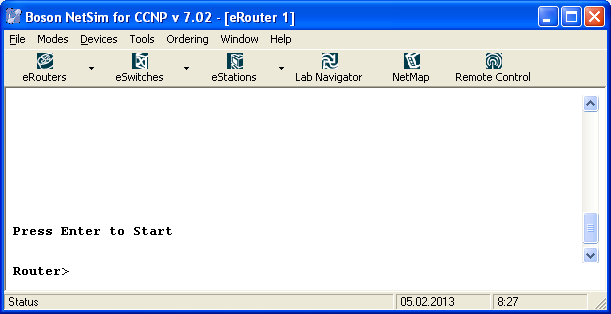


Рис. 11.11 CLI для маршрутизатора Router 1.

C помощью пунктов меню «Devices» и команд Toolbar «eRouters», «eSwitches» и «eStations» можно выбрать конкретное устройство для конфигурирования.

С помощью пунктов меню «File» можно выполнить следующие действия (см. Рис. 11.12):

– New NetMap – создание новой топологии;

– Load NetMap – загрузка готовой топологии из файла;

– Paste Real Router Config – загрузка конфигурации марщрутизатора с реального устройства;

– Load Single Device Config (merge) – загрузка конфигурации отдельного устройства объединением с существующей его конфигурацией;

– Load Single Device Config (overwrite) – загрузка конфигурации отдельного устройства «поверх» существующей конфигурации;

– Load Multi Devices Configs – загрузка конфигурации нескольких устройств;

– Save Single Device Config – сохранение конфигурации отдельного устройства;

– Save Multi Devices Configs – сохранение конфигурации всех устройства топологии.

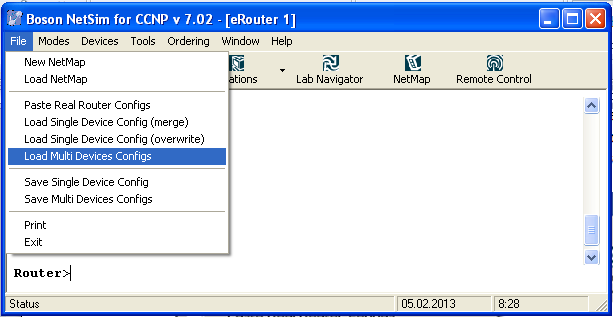


Рис. 11.12 Доступные пункты меню «File».

**Задание**

1. Изучить функциональные возможности программного продукта Boson Network Designer по проектированию сети.
2. Реализовать сеть с топологией, представленной на рис. 11.13. Конкретные модели оборудования необходимо выбрать исходя из наличия у них всех необходимых интерфейсов для соединения.

**Обратить внимание:**

* **на тип устройства: DCE или DTE для соответствующих интерфейсов при построении топологии;**
* **на тип рабочей станции, подключенной к коммутатору Switch2 – TFTP-сервер.**

1. Загрузить рабочую топологию сети, созданную на предыдущей лабораторной работе по рисунку 11.13. Научиться сохранять различными способами конфигурации оборудования для последующего использования.

**ВАЖНО!!!**

После завершения выполнения лабораторной работы файл с топологией и измененные конфигурации оборудования (маршрутизаторов и т.д.) необходимо сохранить для следующих лабораторных работ, чтобы потом не восстанавливать все сначала!!!

**Как сохранить и потом загрузить конфигурацию в Boson NetSim:**

1. **Загружаем топологию из файла – меню «File», далее пункт меню «Load NetMap» (файл топологии уже должен быть сделан).**
2. **Сохраняем конфигурацию – меню «File», далее пункт «Save Multi Devices Configs» и указываем имя файла, желательно в отдельном каталоге (что бы потом не мучиться и не искать нужные файлы), т.к. вместе с общим файлом создаются отдельные файла для каждой «железки» в Вашей сети.**
3. **Перед загрузкой конфигурации необходимо предварительно загрузить соответствующую топологию сети (меню «File» пункт «Load NetMap»).**
4. **Загружаем конфигурацию – меню «File» пункт «Load Multi Devices Configs» и указываем общий файл конфигурации (в этом же каталоге должны лежать и отдельные файла устройств сети).**

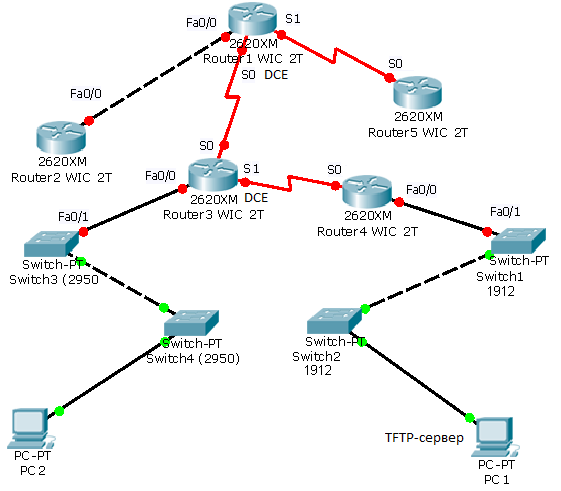


Рис. 11.13 Топология сети.

Таблица IP-адресов и масок интерфейсов представлена ниже (потребуется в дальнейшем при выполнении лабораторных работ, сейчас можно не обращать внимания, но запомнить, что такая табличка есть).

Таблица 11.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **device** | **interface** | **ip address** | **net mask** |
| Router1 | Fa0/0 | 160.10.1.1 | 255.255.255.0 |
| S0 | 175.10.1.1 | 255.255.255.0 |
| S1 | 215.10.1.1 | 255.255.255.0 |
| Router2 | Fa0/0 | 160.10.1.2 | 255.255.255.0 |
| Router3 | S0 | 175.10.1.2 | 255.255.255.0 |
| S1 | 180.10.1.1 | 255.255.255.0 |
| Fa0/0 | 197.10.1.1 | 255.255.255.0 |
| Router4 | Fa0/0 | 195.10.1.1 | 255.255.255.0 |
| S0 | 180.10.1.2 | 255.255.255.0 |
| Router5 | S0 | 215.10.1.2 | 255.255.255.0 |
| Switch1 | - | 195.10.1.99 | 255.255.255.0 |
| Switch2 | - | 195.10.1.100 | 255.255.255.0 |
| Switch3 | - | 197.10.1.99 | 255.255.255.0 |
| Switch4 | - | 197.10.1.100 | 255.255.255.0 |
| PC1 | - | 195.10.1.2 | 255.255.255.0 |
| PC2 | - | 197.10.1.2 | 255.255.255.0 |

# Лабораторная работа 12 Базовая конфигурация маршрутизатора

**Цель**

Изучить основные команды конфигурации маршрутизатора. Выполнить пошагово конфигурацию маршрутизатора.

**Теоретические сведения**

Маршрутизатор, с точки зрения аппаратных и программных компонентов, похож на обычный компьютер, у него также имеются центральный процессор (CPU), Random-Access Memory (RAM), Read-Only Memory (ROM) и операционная система (Internetwork Operating System - IOS).

Центральный процессор (CPU) выполняет инструкции операционной системы, функции маршрутизации и коммутации.

RAM является энергозависимой памятью и ее содержание теряется, когда маршрутизатор будет обесточен или будет выполнена его перезагрузка. В RAM хранятся инструкции и данные, которые используются центральным процессором. RAM используется для хранения:

– операционной системы (IOS); Cisco IOS копируется в RAM в процессе загрузки.

– Running Configuration File (текущая конфигурация), набор команд, которые использует маршрутизатор в настоящий момент; так же называется «running-config».

– IP Routing table (таблица маршрутизации); содержит сведения о непосредственно подключенных и удаленных сетях; используется для определения «лучшего» маршрута для переправления пакетов.

– ARP Cache; содержит сведения о сопоставлении IP адресов и физических MAC адресов; используется на маршрутизаторах, у которых есть интерфейсы, принадлежащие ЛВС (LAN).

– Packet Buffer; используется для временного хранения пакетов, когда они поступают на входящий интерфейс маршрутизатора, и до того момента, когда они покидают исходящий интерфейс.

ROM – это энергонезависимая память, которая используется для хранения:

– команд начальной загрузки;

– диагностического программного обеспечения (используется на этапе загрузки)

Flash Memory (флэш-память) – энергонезависимая память, содержание которой может быть перезаписано; используется для постоянного хранения операционной системы (Cisco IOS); в большинстве маршрутизаторов Cisco операционная система постоянно храниться во флэш-памяти и копируется в RAM в процессе загрузки.

NVRAM – энергонезависимая память; используется в качестве постоянного хранилища конфигурации; сохраняет свое содержимое когда маршрутизатор выключен или была выполнена его перезагрузка.

В процессе выполнения базовой конфигурации маршрутизатора выполняются следующие задачи:

– задание имени;

– установка паролей;

– конфигурация интерфейсов;

– сохранение изменений;

– поверка конфигурации.

В целях безопасности интерпретатор команд маршрутизаторов Cisco предполагает два уровня доступа к командам:

–  пользовательский режим (user EXEC mode); обеспечивает решения задач просмотра состояния маршрутизатора;

– привилегированный режим (privileged EXEC mode); позволяет получить доступ ко всем командам, в том числе командам конфигурации.

Маршрутизаторы имеют специальные физические разъемы, которые используются для управления маршрутизатором. Эти разъемы называются портами управления (console port, консольный порт). В отличие от Ethernet и Serial интерфейсов, консольный порт не используется для пересылки пакетов. Консольный порт используется для подключения терминала, или компьютера со специальным программным обеспечением – эмулятором терминала для настройки маршрутизатора без использования сети. В случае первого включения маршрутизатора его конфигурацию можно выполнить только через консольный порт. Еще одним портом управления является auxiliary порт, не все маршрутизаторы его имеют. Auxiliary порт может быть использован так же как и консольный порт или может использоваться для подключения модема.

Маршрутизаторы имеют еще один вид специальных физических разъемов, основной целью которые является получение и перенаправление пакетов, они называются интерфейсами. Посредством интерфейсов марщрутизатор подключается к нескольким сетям и связывает их. Маршрутизатор может иметь интерфейсы различных типов. Для подключения к локальной сети (LAN), как правило, используются Etharnet или FastEthernet интерфейсы. Для подключения к глобальной сети (WAN) используются Serial, ISDN, Farme Relay. Каждый интерфейс принадлежит определенной сети и Cisco IOS не позволяет двум активным интерфейсам маршрутизатора принадлежать одной и той же сети. Возможна ситуация когда отдельный интерфейс используется для подключения нескольких сетей (виртуальные субинтерфейсы).

Из привилегированного режима EXEC можно перейти в режим «глобальной» конфигурации (это Вы делали в предыдущей лабораторной работе) с помощью команды **configure terminal**. Из режима «глобальной» конфигурации можно получить доступ к нескольким специальным режимам конфигурации:

– Конфигурация интерфейса. Обеспечивает конфигурацию отдельного интерфейса.

– Конфигурация субинтерфейса. Обеспечивает конфигурацию нескольких виртуальных интерфейсов функционирующих на базе одного физического интерфейса.

– Конфигурация контроллера. Обеспечивает конфигурацию контроллеров (например, контроллеров E1 и T1).

– Конфигурация линий. Обеспечивает конфигурацию линий терминала, например, консольных портов и портов VTY (для удаленного подключения).

– Конфигурация протоколов динамической маршрутизации. Обеспечивает конфигурацию протоколов динамической маршрутизации.

**Команды базовой конфигурации**

|  |  |
| --- | --- |
| Router>**enable** | Переход в privileged EXEC mode («привилегированный» режим) |
| Router#**disable** | Выход из privileged EXEC mode («привилегированный» режим) |
| Router#**configure terminal** | Переход в режим «глобальной» конфигурации маршрутизатора |
| Router (config)#**hostname** *name* | Устанавливаем имя маршрутизатору |
| Router (config)#**enable** *password* | Устанавливаем пароль для перехода в privileged EXEC mode («привилегированный» режим); пароль НЕ будет зашифрован и будет отображаться при просмотре конфигурации |
| Router (config)#**enable secret** *password* | Устанавливаем пароль для перехода в privileged EXEC mode («привилегированный» режим); пароль будет зашифрован и не будет отображаться при просмотре конфигурации |
| Router (config)#**interface** *type number* | Переходим в режим конфигурации интерфейса; *type* – тип интерфейса (Ethernet, FastEthernet, Serial); *number*  - номер интерфейса возможно с указанием номера слота и номера интерфейса в слоте (FastEthernet 0/0). |
| Router (config-if)#**ip address** *address mask* | Устанавливаем IP адрес для интерфейса и сетевую маску. |
| Router (config-if)#**description** *description* | Устанавливаем комментарий для интерфейса (например, Link to Router1 WAN). |
| Router (config-if)#**no shutdown** | Переводим интерфейс в «активное» состояние с т.з. физического уровня модели OSI (включаем интерфейс). |
| Router (config)#**copy running-config startup-config** | Сохраняем рабочую конфигурацию в качестве загрузочной конфигурации в NVRAM. |
| Router (config)#**exit** | Переходим в privileged EXEC mode из режима «глобальной» конфигурации (можно использовать комбинацию клавиш Ctrl-Z). |
| Router #**show running-config** | Выводим «рабочую» конфигурацию маршрутизатора. |
| Router #**show startup-config** | Выводим загрузочную конфигурацию маршрутизатора. |
| Router#**show ip interface brief** | Выводим краткую информацию о состоянии интерфейсов маршрутизатора. |
| Router#**show ip interfaces** | Выводим расширенную информацию о интерфейсах маршрутизатора. |
| Router#**show version** | Выводим версию Cisco IOS. |
| Router#**logout** | Завершаем сеанс работы с маршрутизатором. |

**Дополнительные команды**

|  |  |
| --- | --- |
| Router (config)#**line console 0** | Переходим в режим конфигурации консольного порта. |
| Router (config-line)#**password** *password* | Устанавливаем пароль на подключение через консольный порт. |
| Router (config-line)#**login** | Позволяем выполнить процедуру авторизации при подключении через консольный порт. |
| Router (config)#**line vty 0 4** | Переходим в режим конфигурации линий VTY для удаленного подключения. Как и для консольного порта, также необходимо установить пароль (команда password) и разрешить устанавливать удаленные подключения через эти линии (команда login). |
| Router (config)#**banner motd #** *message* **#** | Устанавливаем приветствие при, которое будет отображаться при подключении через терминал. |
| Router (config)#**ip host** *name ip-address* | Устанавливаем соответствие IP-адреса другого устройства сети и его имени для удобства дальнейшей работы. |
| Router#**show hosts** | Выводим таблицу установленных ассоциативных имен устройств сети. |
| Router#**show flash** | Выводим сведения о состоянии flash памяти маршрутизатора. |
| Router#**show history** | Выводим список последний 10 использованных команд. |
| Router(config-if)#**bandwidth** *bandwidth* | Устанавливаем полосу пропускания для интерфейса (не влияет на реальную полосу пропускания интерфейса, а используется в работе некоторых протоколах динамической маршрутизации). |
| Router(config-if)#**clock rate** *value* | Устанавливаем значение тактового генератора (используется для Serial интерфейсов DCE устройств). |

1. Для выполнения работы необходимо загрузить созданную ранее топологию (см. Рис. 11.13).
2. Выберите Router1 из основного меню (пункт «eRouters») и нажмите Enter для получения приглашения ввода команд маршрутизатора.

**Router>**

1. Наберите **?** для получения списка команд доступных в пользовательском режиме (обозначается символом >)

**Router> ?**

1. Введите **enable** для перехода в привилегированный режим (privileged EXEC mode), обозначается символом **#**.

**Router> enable**

**Router#**

1. Сравните список доступных команд в различных режимах (например, команды **Configure** и **Reload** доступны только в привилегированном режиме).
2. Для выхода из привилегированного режима используется команда **Disable**.

**Router# disable**

**Router>**

1. Перейдите снова в привилегированный режим и введите команду **configure terminal**. Таким образом, осуществляется переход в режим конфигурирования маршрутизатора.

**Router> enable**

**Router# configure terminal**

**Router(config)#**

1. Для изменения имени маршрутизатора используется команда **hostname**.

**Router(config)# hostname router1**

**Router1(config)#**

1. Для удобства дальнейшей работы отключаем DNS lookup используя, команду **no ip domain-lookup**. Команда **no ip domain-lookup** отключает в маршрутизаторе преобразование имен в адреса.

**Router1(config)#no ip domain-lookup**

**Router1(config)#**

В режиме конфигурации можно задать два пароля для перехода в привилегированный режим – первый не шифрованный и второй шифрованный. Соответственно первый пароль храниться в открытом виде и его можно будет увидеть при просмотре текущей (running config) конфигурации устройства, а второй пароль будет зашифрован и его нельзя будет подсмотреть в дальнейшем. Понятно, что шифрованный пароль предпочтителен с точки зрения безопасности. Прежде всего, командой **enable ?** посмотрите доступные параметры и формат самой команды и затем задайте два пароля.

**Router1(config)# enable ?**

**Router1(config)# enable password lab**

**Router1(config)# enable secret ciscolab**

1. Простейшая конфигурация интерфейсов предполагает задание IP адреса и сетевой маски. При именовании интерфейсов используется следующий формат записи:

<тип интерфейса><номер слота>/<номер интерфейса в слоте>

Задайте IP адрес для интерфейса **FastEthernet0/0** и активируйте его (изначально интерфейс находится в «выключенном состоянии») для этого используются следующие команды:

**Router1(config)# int *( и жмем клавишу <tab>)***

**Router1(config)# interface FastEthernet0/0**

**Router1(config-if)# ip address 160.10.1.1 255.255.255.0**

**Router1(config-if)# no shutdown**

Очень удобно использовать клавишу <Tab> для подсказки возможных вариантов продолжения названия команд.

Обратите внимание, как изменилась строка приглашения, когда вошли в режим конфигурирования интерфейса. IP адрес необходимо взять из общей таблицы адресов и масок интерфейсов для данной топологии сети (см. Таблицу 11.1).

1. Теперь задайте IP адрес и активируйте интерфейс **Serial0**, используя при этом сокращенные варианты команд

**Router1(config-if)# int s0**

**Router1(config-if)# ip address 175.10.1.1 255.255.255.0**

**Router1(config-if)# no shut**

1. Выйдите из режима конфигурирования интерфейса, используя комбинацию **ctrl-z**

**Router1(config-if)# *ctrl-z***

**Router1#**

Обратите внимание, что произошел выход не только из режима конфигурации интерфейса, но и из режима «глобальной» конфигурации.

1. Закройте сеанс работы с маршрутизатором с помощью команды **logout**.

**Router1# logout**

1. Нажмите Enter и откройте сеанс работы с маршрутизатором получив приглашение вводить команды (помним, что пока находимся в режиме простого пользователя).

**Router1>**

1. Переходим в привилегированный режим

**Router1> enable**

Если при конфигурации были сконфигурированы и «enable password» и «secret password», то именно последний необходимо использовать для перехода в privileged EXEC mode.

1. Просмотрим краткую информацию по интерфейсам

**Router1# show ip interface brief**

1. Просмотрим более детальную информацию по интерфейсам

**Router1# show interfaces**

1. Существует два типа конфигурации операционной системы Cisco ISO: рабочая (running configuration) и загрузочная (startup configuration), часто первую называю активной – она располагается в памяти (RAM) и определяет текущие настройки. Загрузочная конфигурация расположена в энергонезависимой памяти NVRAM и хранит настройки и команда, которые выполняются при загрузке Cisco ISO.
2. Смотрим рабочую конфигурацию

**Router1# show running-config**

1. Смотрим загрузочную конфигурацию.

**Router1# show startup-config**

1. Для записи рабочей конфигурации в NVRAM используется команда **copy**. Таким образом, все текущие изменения конфигурации будут доступны после перезегрузки маршрутизатора.

**Router1# copy running-config startup-config**

1. После записи снова посмотрите загрузочную конфигурацию, там должны быть отражены ваши действия по настройке маршрутизатора.

**Router1# show startup-config**

1. Для просмотра версии операционной системы Cisco ISO и дополнительной информации воспользуйтесь командой:

**Router1# show version**

1. Зададим пароль на консольный вход на маршрутизатор. Все действия проделываем в режиме «глобальной» конфигурации (да это видно по приглашениям, если помните)

**Router1(config)# line console 0**

**Router1(config-line)# login**

**Router1(config-line)# password ваш пароль**

1. Установим сообщение для приветствия при подключении и соответственно перед вводом пароля

**Router1(config)# banner motd #** ***жмем <Enter>***

Super Route Only Super Users ***вводим текст приветствия***

# ***жмем <Enter>***

Символ # вводится перед и после ввода текста приветствия.

1. Оттестируйте результат работы самостоятельно – завершите сеанс и войдите заново

**Router1# logout**

***жмем <Enter>***

1. Для Router2 установим пароль для входа посредством Telnet. Предварительно попадем в режим «глобальной» конфигурации (сами знаете как ☺).

**Router2(config)# line vty 0 4**

**Router2(config-line)# login**

**Router2(config-line)# password cisco**

1. На Router1 пропишем ассоциативное имя соответствующее IP адресу Router2 для удобства работы.

**Router1(config)# ip host router2 160.10.1.2**

1. Проверим результаты труда – посмотрим таблицу хостов на Router1.

**Router1# show hosts**

1. Попробуем «пингонуть» Router2 с Router1 по имени.

**Router1# ping router2**

Если пинг прошел, значит все получилось.

1. Посмотрим информацию о flash памяти на Router2.

**Router2# show flash**

1. Посмотрим последние 10 команд.

**Router1# show history**

Комбинация ctrl-p позволяет увидеть предыдущую команду.

1. На маршрутизаторе Router1 попробуем немного переконфигурировать интерфейс **Serial 0**. При просмотре информации об этом интерфейсе обратите внимание на третью строку, где указан параметр **bandwidth** 1544kb. Маршрутизатор предполагает для всех интерфейсов Serial тип канала T1. Если мы хотим иначе, то придется перенастраивать (например, на 64kb). Кроме этого необходимо настроить параметр **clock rate** (значение тактового генератора) для успешной синхронизации двух связанных маршрутизаторов по Serial интерфейсам.

**Router1# show interfaces serial 0**

**Router1# configure terminal**

**Router1(config)# interface serial 0**

**Router1(config-if)# bandwidth 64**

**Router1(config-if)# clock rate 64000**

**Router1(config-if)# ctrl-z**

**Router1# show interfaces serial 0**

1. И что бы не запутаться с чем соединяется интерфейс, зададим краткое пояснение, что это link к маршрутизатору № 3

**Router1(config)# interface serial 0**

**Router1(config-if)# description Serial Link to Router3**

**Router1(config-if)# exit**

**Router1(config)# exit**

**Router1# show interfaces serial 0**

**Задание**

**1. Проведите настройку второго маршрутизатора (Router2) самостоятельно:**

– **смените имя;**

– **установите пароль для перехода в привилегированный режим;**

– **настройте интерфейс FastEthernet 0/0;**

– **просмотрите информацию по интерфейсам;**

– **проверьте связь между первым и вторым маршрутизаторами (используйте команду ping *ip-address*).**

**2. Проведите настройку маршрутизаторов 3 и 4 самостоятельно:**

– **задайте имя;**

– **задайте secret пароль для привилегированного режима;**

– **задайте IP адреса и маски для интерфейсов в соответствии с таблицей (см. Таблицу 1.);**

– **на маршрутизаторе 3 установите для интерфейса Serial 1 (тот, что соединен с Router4) параметр bandwidth = 64kb и clock rate);**

– **активируйте сконфигурированные интерфейсы;**

– **проверьте связь Router3 с Router1 и Router4;**

– **запишите конфигурацию в NVRAM.**

# Лабораторная работа 13 Статическая маршрутизация.

**Цель**

Изучить основные команды и последовательность действий при настройке статической маршрутизации.

**Теоретические сведения**

Основной функцией маршрутизаторов является маршрутизация информации из одной сети в другую. В маршрутизаторах есть таблица маршрутизации, соответственно которой маршрутизируются данные устройством. В настоящее время существует два способа задания данной маршрутизации:

- Протоколы маршрутизации.

- Статическая маршрутизация.

Протоколы маршрутизации позволяют маршрутизаторам динамически узнавать информацию о путях из одной сети в другую. Вот некоторые из них - RIP, EIGRP, BGP, OSPF, IS-IS.

Во время статической маршрутизации администратор вручную прописывает пути из одной сети в другую. Для того, чтобы вручную прописать маршрут из одной сети в другую в маршрутизаторах CISCO есть команда **ip route**. Синтаксис этой команды следующий:

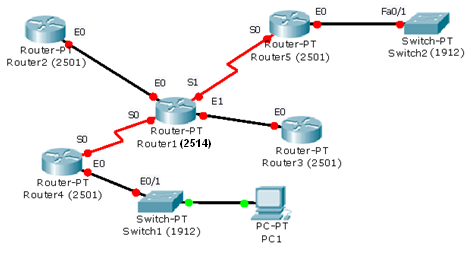
**ip route <*destination ip network address*> <*mask*> <*interface/next hop ip address*>.**

В случае, если маршрутизатор не может найти в своей таблице маршрутизации необходимой записи и не знает ничего о сети, в которую необходимо направить данные, то тогда роутер отправляет данные на шлюз по умолчанию. Для того, чтобы задать шлюз по умолчанию используется следующая команда:

**ip route 0.0.0.0 0.0.0.0 <*interface/ next hop ip address*>.**

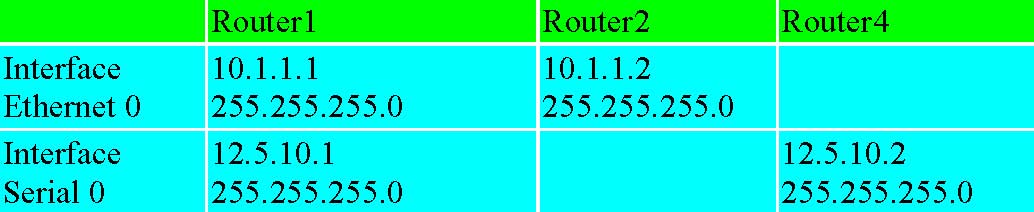
Для того чтобы посмотреть таблицу маршрутизации, в CISCO используется команда **show ip route**.

1. Для выполнения работы необходимо создать новую топологию (см. рисунок ниже).



2. При выполнении работы будет выполняться конфигурация маршрутизаторов Router1, Router2 и Router4 для обеспечения связи между ними.

3. Задайте имена маршрутизаторов и сконфигурируйте интерфейсы (см. Таблицу).



4. Проверьте связь между непосредственно подключенными интерфейсами с помощью команды **ping**.

5. Рассмотрим маршрутизатор Router1. Конфигурация статических маршрутов для него не требуется, т.к. Router2 и Router4 непосредственно подключены к нему.

6. На маршрутизаторе Router4 необходимо настроить статический маршрут до Router2, т.к. они могут связываться между собой только через Router1 или через сеть 10.1.1.0. Для задания статического маршрута к сети 10.1.1.0 используем команду:

**Router4(config)#ip route 10.1.1.0 255.255.255.0 12.5.10.1**

Таким образом, пакеты предназначенные для сети 10.1.1.0 будут направляться через IP адрес 12.5.10.1, который назначен интерфейсу **Serial0** **Router1**.

7. Последовательно проверим доступность интерфейсов на Router 1 - Serial 0 и Ethernet 0, на Router 2 - Ethernet 0.

**Router4#ping 12.5.10.1**

**Router4#ping 10.1.1.1**

**Router4#ping 10.1.1.2**

8. Почему нет ответа от 10.1.1.2, Router4 знает маршрут до Router2 (или сети 10.1.1.0), в этом можно убедиться посмотрев таблицу маршрутизации с помощью команды **show ip route**.Дело в том, что Router2 не имеет маршрута до Router4 (не настроили мы его еще), поэтому он не знает как ответить на ICMP запрос.

**Router4#show ip route**

9. Задайте статический маршрут для Router2 в сеть 12.5.10.0

**Router2(config)#ip route 12.5.10.0 255.255.255.0 10.1.1.1**

10. Проверьте доступность router2 с Router4.

**Router4#ping 12.5.10.1**

**Router4#ping 10.1.1.1**

**Router4#ping 10.1.1.2**

11. Убедитесь, что на Router2 существует маршрут в сеть 12.5.10.0.

**Router2#show ip route**

12. Для удаления статического маршрута используйте ключевое слово **no**.

**Router2(config)#no ip route 12.5.10.0 255.255.255.0 10.1.1.1**

**Задание**

**Используя топологию, представленную на Рис. 11.13. (см. лабораторную работу № 11) и таблицу IP адресов (см. Таблица 11.1), настройте статическую маршрутизацию между Router1, Router2, Router3, Router4 и Router5. Любой из указанных маршрутизаторов должен «пинговаться» с любого другого.**

# Лабораторная работа 14 Динамическая маршрутизация (протокол RIP).

**Цель**

Изучить основные команды и последовательность действий при настройке динамической RIP маршрутизации.

**Теоретические сведения**

RIP - один из самых первых протоколов маршрутизации, но он до сих пор используется в небольших сетях. В настоящий момент существуют две версии данного протокола. В качестве метрики (при выборе оптимального маршрута) используется количество переходов до нужной сети (максимальное количество переходов-15). По умолчанию маршруты обновляются каждые 30 секунд. Обновления таблицы маршрутизации отправляются только соседним маршрутизаторам. Протокол RIP версии 1 поддерживает только сети со стандартной маской сети, то есть сети класса A, B, C. RIP версии 2 в свою очередь поддерживает сети, в которых используются не стандартные маски. Так же RIP версии 2 поддерживает аутентификацию.

Для того чтобы войти в режим конфигурирования RIP необходимо набрать команду **router rip**. Далее с помощью команды **version <1|2>** можно указать какую версию протокола вы настраиваете для данного маршрутизатора (рекомендуется всегда использовать RIP v2, так как он полностью совместим с RIP v1). Затем с помощью команд **network <адрес сети>** задаются те напрямую подключенные к маршрутизаторы сети, которые маршрутизатор должен объявлять (сообщать) для других маршрутизаторов в вашей сети.

**Основные команды**

|  |  |
| --- | --- |
| Router(config)#**router rip** | Переход в режим конфигурации RIP маршрутизации. |
| Router(config-router)#**version** *1|2* | Установить версию протокола. |
| Router(config-router)#**network** *ip-address-network* | Устанавливаем адрес сети, сведениями о которой будет обмениваться маршрутизатор. |
| Router#**show ip route** | Выводит таблицу маршрутизации. |
| Router#**show ip protocols** | Выводит сведения о работающих на маршрутизаторе протоколах динамической маршрутизации. |
| Router#**debug ip rip** | Включаем режим отладки работы протокола RIP. |

1. Для выполнения работы необходимо использовать топологию, созданную в лабораторной работе 13 (или создать заново).

2. При выполнении работы будет выполняться конфигурация маршрутизаторов Router1, Router2 и Router4 для обеспечения связи между ними.

3. Задайте имена маршрутизаторов и сконфигурируйте интерфейсы (см. Таблицу).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Router1 | Router2 | Router4 |
| Ethernet 0 Interface | 10.1.1.1  255.255.255.0 | 10.1.1.2  255.255.255.0 |  |
| Interface Serial 0 | 172.16.10.1 255.255.255.0 |  | 172.16.10.2 255.255.255.0 |

4. Проверьте связь между непосредственно подключенными интерфейсами с помощью команды **ping**.

5. Для настройки RIP маршрутизации необходимо, во-первых, включить ее и, во-вторых, прописать сети, непосредственно подключенные к маршрутизатору (как и было указано выше). После обмена routing информацией в таблицах маршрутизации будут созданы записи соответствующих маршрутов. Командой router rip в режиме «глобальной» конфигурации включаем использование RIP маршрутизации и с помощью команды **network** задаем подключенные сети.

**Router1# Router1#configure terminal**

**Router1(config)#**

**Router1(config)#router rip**

**Router1(config-router)#**

**Router1(config-router)#network 10.0.0.0**

**Router1(config-router)#network 172.16.0.0**

6. Аналогичные действия проделываем на Router2 и Router4

**Router2#**

**Router2#config Terminal**

**Router2(config)#**

**Router2(config)#router rip**

**Router2(config-router)#**

**Router2(config-router)#network 10.0.0.0**

**Router4#**

**Router4#config Terminal**

**Router4(config)#**

**Router4(config)#router rip**

**Router4(config-router)#**

**Router4(config-router)#network 172.16.0.0**

7. Проверим связь между Router2 и Router4

**Router2#ping 172.16.10.2**

**Router4#ping 10.1.1.2**

8. Посмотрите таблицу маршрутизации с помощью следующей команды

**Router1#show ip route**

9. Детальную информацию о работающих на устройстве протоколах маршрутизации можно посмотреть с помощью следующей команды

**Router1#show ip protocols**

**Задание**

**Используя топологию, представленную на Рис. 11.13 и таблицу IP адресов (см. Таблица 11.1), настройте динамическую RIP маршрутизацию между Router1, Router2, Router3, Router4 и Router5. Предварительно удалите статические маршруты, если они остались прописаны на маршрутизаторах. Любой из указанных маршрутизаторов должен «пинговаться» с любого другого.**

# Лабораторная работа 15 Комплексная работа по настройке сети.

**Цель**

Закрепление навыков конфигурирования сетевых устройств.

**Задание**

1. Реализовать представленную на рисунке топологию сети, использовать маршрутизаторы 2621 и коммутаторы 2950.

2. Выполнить базовую настройку сетевых устройств:

– задать имя;

– установить пароль для входа через консольный порт;

– установить пароль для перехода в привилегированный режим;

– настройте приветствие (баннер) при входе;

– обеспечьте возможность подключения к маршрутизаторам с помощью Telnet;

– задать значимые комментарии к интерфейсам

3. Выполните эффективное планирование адресного пространства и заполните недостающие данные таблицы:

– во всех LAN не более 62 хостов;

– диапазон для сегмента R1-ISP – 209.165.200.224/27;

– диапазон для сегмента ISP-Web Server – 192.168.99.0/24;

– диапазон для сегментов между маршрутизаторами WAN – 10.1.1.0/24 (соответственно эффективно выделите IP-адреса для трех подсетей WAN);

– диапазон для сегмента LAN, подключенного к R2 – 192.168.20/24;

– сегмента LAN, подключенный к R2 содержит две VLAN: VLAN 10 (PC 1, PC 2, PC 4 и PC 5) и VLAN 20 (PC 3 и PC 6)

– для сегмента LAN, подключенного к R2, выделить три диапозона IP адресов (подсети); первую подсеть использовать для устройств VLAN 10, вторую для VLAN 20, третий и четвертый диапазоны остаются как резервные;

– интерфейсу Fa 0/1 R2 назначить первый IP адрес из диапазона VLAN 10;

–  интерфейсу Fa 0/0 R2 назначить первый IP адрес из диапазона на VLAN 20;

– рабочие станции VLAN 10 и VLAN 20 используют следующие по порядку свободные IP адреса из доступного диапазона;

– диапазон для сегмента LAN, подключенного к R3 – 192.168.30.0/24;

4. Выполните настройку сетевых интерфейсов.

7. Выполнить настройку статической маршрутизации до ISP.

8. Выполнить настройку RIP маршрутизации между маршрутизаторами R1, R2, R3.

9. Обеспечить связь между всеми устройствами в сети.

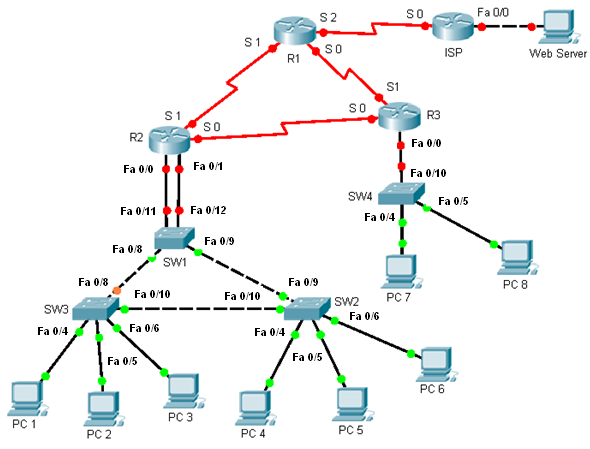


Рис. 15.1 Топология сети

Таблица IP адресов.

|  |  |  |  |
| --- | --- | --- | --- |
| **device** | **interface** | **ip address** | **default gateway** |
| ISP | S0 | 209.165.200.225/27 |  |
| Fa 0/0 | 192.168.99.1/24 |  |
| R1 | S0 |  |  |
| S1 |  |  |
| S2 | 209.165.200.226/27 |  |
| R2 | S0 |  |  |
| S1 |  |  |
| Fa 0/0 |  |  |
| Fa 0/1 |  |  |
| R3 | S0 |  |  |
| S1 |  |  |
| Fa 0/0 |  |  |
| SW1 | Vlan 1 |  |  |
| SW2 | Vlan 1 |  |  |
| SW3 | Vlan 1 |  |  |
| PC 1 | NIC |  |  |
| PC 2 | NIC |  |  |
| PC 3 | NIC |  |  |
| PC 4 | NIC |  |  |
| PC 5 | NIC |  |  |
| PC 6 | NIC |  |  |
| PC 7 | NIC |  |  |
| PC 8 | NIC |  |  |
| Web Server | NIC | 192.168.99.2/24 |  |