Сетевая поддержка в Линуксе, Linux NET-3-HOWTO.

Тэрри Доусон (Terry Dawson) (основной автор), VK2KTJ; Алессандро Рубини (сопровождающий), alessandro.rubini@linux.it Егор Дуда (перевод) deo@logos-m.ru v1.4, Август 1998

Сетевая поддержка в Операционной системе Линукс написана "с нуля" почти целиком. Производительность подсистемы tcp/ip в последних версиях ядра делает ее достойной альтернативой другим реализациям. В этом документе описывается, как установить и настроить сетевую поддержку и связанные с ней утилиты.

Содержание

1 Отличия от версии 1.3 (Апрель 1998)

```
Добавления:
        Ограничитель потока данных (Traffic Shaper).
        Поддержка PLIP в новых ядрах.

Исправления/изменения:
        Адрес сопровождающего NetKit.
        Изменено описание доменной системы имен.
        Реорганизованы разделы.
        Отмечены отличия ядер 2.0 и 2.2 (информация о ядрах 2.2 пока отсутствует)
```

```
        Исправления ссылок.

Планы:
        Описать новый алгоритм маршрутизации.
        Описать опции компиляции ядра для IPv6.
        Описать содержимое каталога /proc/sys/net/
        Устройство WanRouter
        Описать новые команды файрволла в ядрах версий 2.2
```

2 Введение.

Первый NET-FAQ, содержавший ответы на часто задаваемые вопросы по работе с сетью в Линуксе, был написан Маттом Уэлшем (Matt Welsh) и Тэрри Доусоном (Terry Dawson) до того, как был формально начат проект по документированию Линукса. Он описывал самые первые версии сетевой подсистемы Линукса. Затем последовал NET-2-HOWTO, расширивший круг рассматриваемых вопросов по сравнению с NET-FAQ. Он был одним из первых HOWTO, и описывал версию 2 (а в дальнейшем и версию 3) сетевой подсистемы Линукса. Этот документ, в свою очередь, стал расширением NET-2-HOWTO и описывает только версию 3 сетевой подсистемы Линукса.
Предыдущие версии этого документа были довольно объемными из-за большого количества материала, связанного с сетевыми возможностями Линукса. Сейчас специализированные разделы вынесены из этого документа в отдельные HOWTO. Этот документ содержит ссылки на них и описания оставшихся областей.
В апреле 1998 года Тэрри прекратил сопровождение этого документа в связи со своей высокой загруженностью и передал его Алессандро Рубини (Alessandro Rubini).

2.1 Отзывы

Все Ваши отзывы и предложения будут с благодарностью приняты. Отправляйте их по адресу (*rubini@linux.it* <mailto:rubini@linux.it>.

3 Использование этого документа (NET-3-HOWTO-HOWTO ?).

Этот документ имеет иерархическую структуру. Информационные разделы собраны в начале, и Вы можете их пропустить без вреда для понимания остальных разделов. Далее собран общий материал, необходимый для понимания остальных разделов документа. Последняя часть документа, описывающая конкретные сетевые технологии разбита на 3 части: Информация об ethernet- и IP-сетях, технологии, относящиеся к распространенному аппаратному обеспечению, и редко используемые технологии.
Рекомендуем Вам следующий порядок прочтения этого документа:

Прочтите общие разделы

Информация из этих разделов используется практически во всех дальнейших разделах, поэтому убедитесь, что вы хорошо разобрались в ней, прежде чем читать дальше.

Обдумайте структуру вашей сети

Вы должны четко представлять структуру вашей текущей или будущей сети и какие аппаратные средства и сетевые технологии Вы используете или будете использовать.

Прочтите раздел “Ethernet и IP”, если вы подключены к локальной сети с выходом в Интернет.

В этом разделе описаны основные настройки ethernet-сети и различные возможности поддержки IP-сетей в Линуксе, такие как файрволлы, продвинутая маршрутизация и т.д.

Прочтите следующий раздел, если Вас интересуют вопросы сеансовых соединений

В этом разделе описаны протоколы PLIP, PPP, SLIP и ISDN — технологии, распространенные на персональных компьютерах.

Прочтите разделы, относящиеся к вашей сети

Если Вы используете специфическое оборудование или протоколы, отличные от IP, прочтите соответствующие разделы последней части.

Настройте Вашу сеть

На этом этапе Вы можете попытаться настроить Вашу сеть и выяснить, возникают ли какие-либо проблемы в работе сети.

Выясните, куда обращаться за помощью

Если Вы столкнулись с проблемами, которые Вы не смогли решить с помощью этого документа, прочтите раздел, содержащий информацию о том, куда обращаться за помощью и сообщать о найденных ошибках.

Наслаждайтесь!

Получайте удовольствие, работая в сети.

3.1 Обозначения, принятые в этом документе

В данном документе нет особой системы обозначений, единственное, на что следует обратить внимание — способ записи команд. Используя "классический" стиль юникса, каждой команде предшествует приглашение. В этом документе команды, не требующие администраторских привилегий приведены с приглашением вида "`user%`", а требующие — "`root#`"вместо обычного "#". Это сделано для того, чтобы отличить их от комментариев.
В пунктах, озаглавленных "Опции компиляции ядра" используется формат программы *menuconfig*. Надеемся, они будут понятны и тем, кто не пользуется этой программой. Если Вы не уверены в порядке расположения опций, просто запустите программу *menuconfig*.
Все ссылки на другие документы HOWTO сделаны локальными, для того, чтобы Вы могли просматривать HTML-версии с помощью броузера. Если у Вас на машине нет полного набора этих документов, их можно получить с сайта `sunsite.unc.edu` (каталог `/pub/Linux/HOWTO`) или с любого из его зеркал.

4 Общая информация о сетевых средствах Линукса.

4.1 Краткая история разработки сетевой подсистемы Линукса.

Разработка совершенно новой реализации стека tcp/ip, сравнимого по возможностям и производительности с существующими реализациями была нелегкой задачей. Отказ от переноса существующей реализации был сделан в тот момент, когда существовала некоторая неясность, не станут ли существующие реализации отягощены ограничениями авторских прав в связи с судебным делом, начатым U.S.L. Кроме того, существовал значительный энтузиазм сделать все по-своему и лучше, чем делалось ранее.
Первым добровольцем, возглавившим разработку сетевой подсистемы был Росс Биро (Ross Biro <`biro@yggdrasil.com`>). Росс написал простую и неполную, но уже пригодную к использованию систему, дополненную драйвером для сетевой карты ethernet WD-8003. Этого было достаточно, чтобы многие могли экспериментировать и тестировать новую систему, а некоторые даже смогли с ее помощью подключить свои машины к интернету. Давление, которое сообщество разработчиков Линукса, оказывало на разработчиков сетевой подсистемы было весьма большим и в определенный

момент вынудило Росса отказаться от руководства проектом. Но его усилия по начальной организации проекта и взятая ответственность за создание чего-либо работоспособного в сложных условиях стали катализатором для всех дальнейших работ и заложили основы современного успеха.

Орест Збровски (Orest Zborowski <obz@Kodak.COM>) написал первый вариант интерфейса BSD sockets. Это был значительный шаг вперед, так как это позволило переносить многие существовавшие сетевые приложения без серьезных модификаций.

Примерно в это-же время Лоуренс Калхейн (Laurence Culhane <loz@holmes.demon.co.uk>) разработал первый драйвер поддержки протокола SLIP. Это позволило многим, не имеющим доступа к локальным ethernet-сетям, экспериментировать с новым программным обеспечением. И снова, некоторым удалось с помощью этого драйвера подключить свои машины к интернету. Это дало многим ощущение возможностей, которые будет иметь Линукс с полной сетевой поддержкой и увеличило число людей, экспериментирующих с сетевой подсистемой.

Еще одним из активных разработчиков был Фред ван Кемпен (Fred van Kempen <waltje@uwalt.nl.mugnet.org>). После краткого периода неопределенности, последовавшего за уходом Росса, Фред принял на себя руководство проектом, без особой конкуренции. У Фреда были свои планы развития сетевой подсистемы Линукса, и он направил основную работу именно в этих направлениях. Под его руководством были написаны программы, названные NET-2 (в отличие от NET — программ, написанных Россом), которые многие смогли продуктивно использовать. Кроме того, Фред внес много предложений по новым разработкам, таким как динамический интерфейс устройства, поддержка протокола Amateur Radio AX.25 и модульная структура сетевой подсистемы. NET-2 использовался большим количеством людей, которое постоянно увеличивалось по мере того, как распространялась информация о работоспособности этой системы. В тот момент новая реализация все еще выпускалась как набор исправлений к обычному ядру и не включалось в основное выпускаемое ядро. NET-FAQ и NET-2-HOWTO описывали достаточно сложную процедуру, которая требовалась, чтобы заставить сетевую подсистему работать. Фред уделял основное внимание новым разработкам, что отнимало основную часть времени. В то же время пользователям требовалась надежная система, которая устроила бы по крайней мере 80% пользователей. Так же, как и в случае с Россом, пользователи стали оказывать на разработчиков давление.

Алан Кокс (Alan Cox <iialan@www.uk.linux.org>) предложил решить возникшие проблемы следующим образом. Он предложил взять на себя отладку кода NET-2 до состояния стабильной и надежной работы, которая бы устроила большинство пользователей. При этом давление на Фреда уменьшилось бы и позволило бы ему продолжить свою работу. Алан взялся за работу и через некоторое время выпустил первую версию сетевой подсистемы NET-2D (от NET-2-Debugged). Она устойчиво работала на большинстве машин, и удовлетворяла большинство пользователей. У Алана были свои взгляды на то, как должен развиваться проект и это привело к большому количеству дискуссий о том, как развивать NET-2. В результате сложилось два подхода среди разработчиков сетевой подсистемы — первый — "сперва работоспособность, затем доводка"и второй — "доводка в процессе разработки". Линус Торвальдс (Linus Torvalds) выступил в качестве арбитра и поддержал Алана, включив его код в стандартное выпускаемое ядро. Фред оказался в сложной ситуации. Все его дальнейшие разработки лишались большого количества тестеров, а это означало замедление прогресса. Фред работал еще некоторое время, а затем Алан стал новым лидером команды разработчиков сетевой подсистемы.

Дональд Беккер (Donald Becker <becker@cesdis.gsfc.nasa.gov>) занялся программированием нижнего уровня сетевой подсистемы и написал огромное количество драйверов ethernet-карт. Почти все драйвера, включенные в текущие версии ядра написаны Дональдом. В написании драйверов участвовали и другие, но работа Дональда была столь продуктивной, что заслуживает особого упоминания.

Алан продолжал улучшать NET-2D, параллельно занимаясь проблемами, не указанными явно в списке первоочередных. К тому моменту, когда версии основного ядра 1.3.* достигли состояния зрелости, сетевая подсистема переросла в версию 3 — NET-3, на базе которой основана текущая реализация сетевой поддержки в Линуксе. Алан работал над многими частями сетевой подсистемы и с помощью многих других развивал ее по многим направлениям. Алан разработал динамические сетевые устройства и первые реализации протоколов AX.25 и IPX. Параллельно Алан продолжал

дорабатывать остальной код и продолжает заниматься этим и сейчас.
Поддержка PPP была написана Майклом Коллахэном (Michael Callahan <callahan@maths.ox.ac.uk>) и Элом Лонгйиаром (Al Longyear <longyear@netcom.com>). Это также имело огромное значение, так как значительно увеличило число пользователей Линукса.
Джонатан Нейлор (Jonathon Naylor <jsn@cs.nott.ac.uk>) значительно улучшил поддержку AX.25, добавив поддержку протоколов NetRom и Rose. Поддержка AX.25/NetRom/Rose имела очень большое значение, поскольку ни одна операционная система кроме Линукса не имела встроенной поддержки этих протоколов.
Конечно, многие сотни людей участвовали в разработке сетевой подсистемы Линукса. Многие из них будут упоминаться в специальных разделах, многие обеспечили написание модулей и драйверов, высказывали пожелания, присылали сообщения об ошибках и оказывали моральную поддержку. Все они сыграли свою роль в разработке. Сетевая подсистема Линукса — отличный пример результата, достигнутого "анархической"разработкой, о она продолжает развиваться по многим направлениям и сейчас.

4.2 Источники информации о сетевой подсистеме Линукса.

Информацию о работе с сетью в Линуксе Вы можете почерпнуть из множества источников.
Алан Кокс, нынешний ведущий разработчик сетевой подсистемы Линукса сопровождает web-страницу, на которой содержатся все последние новости, касающиеся сетевой поддержки в Линуксе. Эта страничка находится по адресу *www.uk.linux.org* <http://www.uk.linux.org/NetNews.html>.
Другой полезный источник — книга Олафа Кирха (Olaf Kirch) "Network Administrator Guide" ("Руководство сетевого администратора"). Она является частью Проекта по Документации Линукса (*Linux Documentation Project* <http://sunsite.unc.edu/LDP/>) и вы можете прочесть ее (на английском языке) HTML-версию по адресу *Network Administrators Guide HTML version* <http://sunsite.unc.edu/LDP/LDP/nag/nag.html> либо получить ее в других форматах по ftp с сайта sunsite.unc.edu *sunsite.unc.edu LDP ftp archive* <ftp://sunsite.unc.edu/pub/Linux/docs/LDP/network-guide/>. Книга Олафа — весьма полное руководство по настройке сетевой поддержки под Линуксом.
Существует news-группа в иерархии news-групп, посвященных Линуксу, в которой обсуждаются вопросы, посвященные сетевой подсистеме — *comp.os.linux.networking* <news:comp.os.linux.networking>
Кроме того существует лист рассылки, подписавшись на который Вы можете задать вопросы, относящиеся к сети под Линуксом. Для того, чтобы подписаться, отправьте письмо:

```
To: majordomo@vger.rutgers.edu
Subject: <любой>

В теле письма напишите:

subscribe linux-net
```

В различных IRC-сетях часто есть каналы #linux в которых Вы можете получит ответы на многие вопросы.
Сообщая о какой либо проблеме, старайтесь включить как можно больше подробностей, имеющих к ней отношение. В частности, обязательно сообщайте номера версий используемых программ, ядра, номера версий таких программ как *pppd* или *dip*, и суть возникшей проблемы. Обращайте особое внимание на все получаемые сообщения об ошибках и точно упомяните все вводимые Вами команды.

4.3 Источники информации о сетях, не связанные непосредственно с Линуксом

Для получения базовой информации о работе tcp/ip сетей советуем Вам обратиться к следующим документам:

tcp/ip introduction

этот документ можно получить как в *текстовом виде* <ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc>, так и в виде *pstscript* <ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps>.

tcp/ip administration

этот документ можно получить как в *текстовом виде* <ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc>, так и в виде *postscript* <ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps>.

За более подробной информацией Вы можете обратиться к книге

Douglas E. Comer *Internetworking with TCP/IP, Volume 1: principles, protocols and architecture*, ISBN 0-13-227836-7 Prentice Hall publications, Third Edition, 1995.

Если Вы хотите писать сетевое программное обеспечение в Unix-подобных средах, очень рекомендуем Вам прочесть книгу

W. Richard Stevens *Unix Network Programming* ISBN 0-13-949876-1, Prentice Hall publications, 1990.

Ожидается к выходу второе издание этой книги, в новом издании она разбита на три тома, подробности на *Web-сайте фирмы Prentice-Hall* <http://www.phptr.com/>.
Кроме того, можете обратиться к news-группе *comp.protocols.tcp-ip* <news:comp.protocols.tcp-ip>.
Важным источником информации и протоколах семейства tcp-ip и интернете являются документы RFC. RFC расшифровывается как "Request For Comment"(Запрос для Обсуждения). RFC — способ описания стандартов Интернет. Набор документов RFC можно получить из многих мест, по протоколу ftp или через WWW. Многие сайты обеспечивают доступ к RFC с возможностью поиска по ключевым словам. Один из таких сайтов находится по адресу: *Nextor RFC database* <http://pubweb.nexor.co.uk/public/rfc/index/rfc.html>.

5 Общая информация по настройке сети.

Понимание следующих подразделов необходимо для конфигурирования сети. Они содержат информацию, необходимую Вам независимо от типа и структуры Вашей сети.

5.1 С чего начать?

Для компиляции и настройки сетевой подсистемы Вам нужны несколько вещей. Самые важные из них:

5.1.1 Исходный код текущего ядра.

Поскольку ядро, используемое Вами сейчас, может не иметь встроенной поддержки для тех типов сетевых карт и тех протоколов, которые Вы захотите использовать, Вам может потребоваться перекомпилировать ядро с соответствующими опциями.

Вы всегда можете получит исходный код последних версий ядра по ftp с *основного сайта ftp.kernel.org* <ftp://ftp.kernel.org/pub/linux/kernel>. Следует отметить, однако, что этот сайт перегружен, поэтому старайтесь получать исправления вместо файлов целиком, кроме того, рекомендуем Вам воспользоваться "зеркалами", такими как *ftp.funet.fi* <ftp://ftp.funet.fi//mirrors/ftp.kernel.org/pub/linux/kernel>. Кроме того, свежие исходные тексты ядра обычно есть на многих сайтах, посвященных Линуксу.
Исходный код ядра обычно распаковывают в каталог `/usr/src/linux` За информацией о том, как внести в ядро исправления и скомпилировать его, обращайтесь к *Kernel-HOWTO* <Kernel-HOWTO.html>. Информация о конфигурировании ядра содержится в "Modules-mini-HOWTO". Еще одним полезным источником информации являются файл `README` и каталог `Documentation`, входящие в состав исходных текстов ядра.
Рекомендуем Вам пользоваться стандартным выпуском ядра (с четной второй цифрой в номере версии), если в тексте явно не сказано обратное. Выпуски ядра для разработчиков (с нечетной второй цифрой в номере версии) обычно имеют серьезные структурные изменения и могут оказаться несовместимыми с Вашими программами. Если вы не уверены в своей способности (или желании) решить подобные проблемы, в дополнение к возможным ошибкам в этих версиях ядра, не используйте их.
С другой стороны, некоторые из описываемых возможностей были введены в ядрах версий 2.1, поэтому у Вас есть выбор — либо пользоваться версиями 2.0 и ждать версий 2.2 со всеми программами поддержки новых возможностей, либо пользоваться версиями 2.1 и искать эти программы самостоятельно. На момент написания этого абзаца (Август 1998) последней версией была 2.1.115 и появление версии 2.2 ожидалось в ближайшем будущем.

5.1.2 Сетевые утилиты.

С помощью этих программ Вы можете конфигурировать сетевые устройства. Они, например, позволяют Вам назначать сетевые адреса и настраивать маршрутизацию.
Большинство современных дистрибутивов Линукса включают эти программы, поэтому, если вы не установили их из вашего дистрибутива, самое время сделать это.
Если Вы устанавливали Линукс не из дистрибутива, Вам потребуется скомпилировать эти программы из исходного кода. Это не слишком сложно.
Набор сетевых утилит ведется сейчас Берндом Экенфельдсом (Bernd Eckenfels) и может быть получен по ftp по адресу *ftp.inka.de* <ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/> или с "зеркала"*ftp.uk.linux.org* <ftp://ftp.uk.linux.org/pub/linux/Networking/base/>.
Убедитесь, что версия полученного Вами пакета совместима с Вашим ядром. После этого следуйте содержащимся в пакете инструкциям по установке.
Чтобы скомпилировать и установить этот пакет на момент написания этого документа требовалось выполнить следующие команды:

```
user% tar xvfz net-tools-1.33.tar.gz
user% cd net-tools-1.33
user% make config
user% make
root# make install
```

Если вы собираетесь установит файрволл или использовать средства IP-маскарада Вам понадобится программа *ipfwadm*. Последнюю версию этой программы Вы можете получить по ftp по адресу *ftp.xos.nl* <ftp:/ftp.xos.nl/pub/linux/ipfwadm>. Убедитесь, что полученная версия совместима в Вашим ядром. Так как работа файрволла в версиях 2.1 изменилась, дальнейшее относится только к ядрам версий 2.0.
Для установки и настройки выполните следующие команды:

```
user% tar xvfz ipfwadm-2.3.0.tar.gz
user% cd ipfwadm-2.3.0
user% make
root# make install
```

Если Вы используете версии 2.2 или поздние версии 2.1, настройка файрволла выполняется без программы *ipfwadm*. В данной версии этого документа новая система настройки файрволла не рассматривается.

5.1.3 Сетевые приложения.

К сетевым приложениям относятся, такие программы как *telnet* и *ftp* и их демоны. Дистрибутивы основных программ этого типа ведутся Дэвидом Холландом (David Holland). Пишите по адресу netbug@ftp.uk.linux.org. Вы можете получить эти программы по адресу *ftp.uk.linux.org* <ftp://ftp.uk.linux.org/pub/linux/Networking/base>.
В марте 1997 года пакет сетевых приложений был разбит на несколько небольших пакетов, а в мае 1997 основные программы были объединены в пакете netkit-base-0.10. Вам может понадобится этот базовый пакет, и некоторые из дополнительных.
Для установки и настройки этих программ выполните следующие команды:

```
user% tar xvfz netkit-base-0.10
user% cd netkit-base-0.10
user% more README
user% vi MCONFIG
user% make
root# make install
```

5.1.4 Адреса.

Адреса в протоколе IP состоят из четырех байт. Их принято записывать в так называемой 'dotted decimal notation' (десятичной нотации). В ней каждый байт представляется десятичным числом (0-255), байты разделены знаками '.' Каждому сетевому интерфейсу на компьютере или маршрутизаторе присваивается IP-адрес. Возможно использование одного адреса несколькими интерфейсами на одной машине, но как правило этого не делают.
IP-сетью называют непрерывную последовательность IP-адресов, такую, что все адреса из последовательности имеют одинаковые старшие биты. Часть адреса, присутствующая во всех адресах последовательности называется сетевой частью' адреса. Оставшаяся часть называется 'машинной частью' Биты, соответствующие сетевой части называется 'маской'. С помощью маски определяют, какие адреса принадлежат сети, а какие — нет. В качестве примера рассмотрим следующую IP-сеть:

```
----------------------  ---------------
Адрес машины            192.168.110.23
Маска                   255.255.255.0
Сетевая часть           192.168.110.
Машинная часть                     .23
----------------------  ---------------
Адрес сети              192.168.110.0
Широковещательный адрес 192.168.110.255
----------------------  ---------------
```

Применяя к адресу и маске операцию 'побитовое И' получаем адрес сети. Таким образом, адрес сети всегда является наименьшим в последовательности и имеет нулевую машинную часть.

'Широковещательный' адрес — специальный адрес, который "слушают"все машины в сети, кроме своего собственного. По этому адресу отправляются пакеты, предназначенные всем машинам в сети. Таким образом передается информация о маршрутизации или сетевые сообщения об ошибках. Есть два стандарта того, какой адрес в сети использовать в качестве широковещательного. Наиболее распространенный стандарт — использовать для этого наибольший адрес в IP-сети. В приведенном примере это `192.168.110.255`. Однако некоторые сайты используют сетевой адрес в качестве широковещательного. На практике не очень важно, какого стандарта придерживаться, однако всегда следует убедится, что все машины в сети сконфигурированы на использование одного широковещательного адреса.
По административным причинам на ранних стадиях разработки протокола IP, некоторые группы адресов были сгруппированы в сети и разбиты на классы. Каждый класс состоит из сетей одинакового объема.

Сетевой Класс	Маска	Сетевые Адреса
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 - 239.255.255.255

Используемые Вами адреса зависят от контекста, в котором Вы их используете. Вам может потребоваться произвести следующие действия:

Установка компьютера в уже существующую сеть

Если Вы собираетесь устанавливать машину с Линуксом в уже существующую сеть, узнайте у Вашего сетевого администратора следующие адреса:

- IP-адрес машины
- IP-адрес сети
- широковещательный IP-адрес
- маску
- IP-адрес маршрутизатора
- IP-адрес сервера имен (DNS-server)

После этого Вы должны сконфигурировать сетевое устройство на вашей машине в на использование этих адресов.

Создание новой сети, которая не будет подключена к интернету.

Если Вы создаете внутреннюю сеть, которую Вы не собираетесь в будущем подключать к интернету, Вы можете выбрать любые адреса. Однако, из соображений безопасности и единообразия, определенные IP-адреса были специально зарезервированы для подобных целей. Эти адреса описаны в RFC1597 следующим образом.

АДРЕСА, ЗАРЕЗЕРВИРОВАННЫЕ ДЛЯ ВНУТРЕННИХ СЕТЕЙ		
Сетевой Класс	Маска	Сетевые адреса
A	255.0.0.0	10.0.0.0 - 10.255.255.255

```
|    B    | 255.255.0.0   | 172.16.0.0  - 172.31.255.255  |
|    C    | 255.255.255.0 | 192.168.0.0 - 192.168.255.255 |
-----------------------------------------------------------
```

Все, что Вам нужно сделать — это решить, сколько адресов Вам нужно, и выбрать подходящий диапазон.

5.2 Где размещать команды конфигурации ?

Есть несколько разных подходов к организации процесса загрузки Линукса. После своей загрузки ядро всегда запускает программу, называющуюся '*init*'. Эта программа читает файл `/etc/inittab` и выполняет процесс загрузки системы. Есть несколько вариантов программы *init*, повсеместно используется версия из System V, написанная Мигелем ван Смуренбургом (Miguel van Smoorenburg.). В разных дистрибутивах процесс загрузки организован по-разному. Обычно файл `/etc/inittab` содержит набор строчек типа

```
si::sysinit:/etc/init.d/boot
```

Эта строчка задает имя скрипта, который выполняет загрузку. Такой скрипт играет роль, схожую с ролью файла `autoexec.bat` в MS-DOS.
Обычно этот скрипт вызывает другие скрипты, и сеть конфигурируется в одном из них.
Следующую табличку можно использовать как руководство:

```
-------------------------------------------------------------------------------
Дистриб. | Настройка интерфейса и маршрутизации       |Запуск демонов
-------------------------------------------------------------------------------
Debian   |/etc/init.d/network                         |/etc/init.d/netbase
         |                                            |/etc/init.d/netstd_init
         |                                            |/etc/init.d/netstd_nfs
         |                                            |/etc/init.d/netstd_misc
-------------------------------------------------------------------------------
Slackware|/etc/rc.d/rc.inet1                          |/etc/rc.d/rc.inet2
-------------------------------------------------------------------------------
RedHat   |/etc/sysconfig/network-scripts/ifup-<ifname>|/etc/rc.d/init.d/network
-------------------------------------------------------------------------------
```

Обратите внимание, что дистрибутивы Debian и Red Hat содержат отдельный каталог для скриптов для запуска системных сервисов (хотя сами файлы настроек находятся в других местах, в дистрибутиве Red Hat они хранятся в каталоге `/etc/sysconfig`). Для понимания процесса загрузки ознакомьтесь с содержимым файла */etc/inittab* и документацией по процессу *init*. Готовится к публикации статья в Linux Journal, как только она будет доступна на www, в этот документ будет включена соответствующая ссылка.
Большинство современных дистрибутивов включают программы для настройки большинства типов сетевых интерфейсов. Если такая программа у вас есть, попробуйте использовать ее, прежде чем вносить исправления вручную.

```
---------------------------------------------
Дистрибутив   | Программа конфигурирования сети
---------------------------------------------
RedHat       | /sbin/netcfg
Slackware    | /sbin/netconfig
---------------------------------------------
```

5.3 Создание сетевых интерфейсов.

Во многих операционных системах из семейства Unix сетевые устройства представлены в виде файлов в каталоге */dev* В Линуксе это не так. В Линуксе сетевые устройства создаются динамически, и поэтому не требуют наличия соответствующих файлов в каталоге */dev*
В большинстве случаев сетевое устройство создается драйвером, после того как тот проинициализируется и обнаружит сетевую карту. Например, драйвер ethernet-карты создает интерфейсы с именами `eth[0..n]` по мере обнаружения всех сетевых карт в Вашей машине. Первая сетевая карта связывается с интерфейсом `eth0`, вторая — `eth1` и т.д.
Однако в некоторых случаях, в частности в случае интерфейсов *slip* и *ppp*, интерфейсы создаются пользовательским программами. При этом сохраняется принцип последовательной нумерации интерфейсов, однако интерфейсы не создаются в момент загрузки. Причина этого состоит в том, что количество работающих *slip*- и *ppp*-интерфейсов может меняться в ходе работы машины, в отличие от количества ethernet-карт. Эти случаи будут подробнее рассмотрены в следующих разделах.

5.4 Настройка сетевого интерфейса.

Итак, у Вас есть все необходимые программы, адреса и информация о сети. Можно приступать к настройке сетевых интерфейсов. Настройка заключается в присвоении соответствующих адресов сетевому устройству и установке нужных значений для других параметров сетевого устройства. Наиболее часто для этого используется программа *ifconfig*.
Вы должны запустить ее примерно следующим образом:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

В этом случае сетевому устройству *eth0* будет присвоен IP-адрес '`192.168.0.1`' и маска '`255.255.255.0`'. Ключ '*up*' в конце команды делает интерфейс активным. Это действие по умолчанию, поэтому Вы можете опустить этот ключ. Для дезактивации интерфейса используйте команду "`ifconfig eth0 down`".
Ядро при конфигурации интерфейса принимает некоторые значения по умолчанию. Например, вы можете явно указать сетевой и широковещательный адрес для интерфейса, однако если Вы этого не сделаете, как в примере выше, ядро попытается "угадать"правильные значения на основе указанных адреса и маски. Если Вы не укажете маску, она будет выбрана в соответствии с тем классом адресов, в который попадет указанный адрес. В примере выше ядро предположило бы, что машина находится в сети класса C, и установило бы сетевой адрес в '`192.168.0.0`' а широковещательный в '`192.168.0.255`'.
Программа *ifconfig* имеет множество опций, наиболее полезные из которых:

up

активирует интерфейс (это действие по умолчанию).

down

дезактивирует интерфейс

[- arp]

разрешает и запрещает использование протокола преобразования адресов (arp) на данном интерфейсе.

[- allmulti]

разрешает и запрещает прием аппаратных пакетов с несколькими адресатами (multicast-пакетов). Такие пакеты позволяют группе машин принимать пакеты, отправленные на специальный адрес. Такая возможность используется в приложениях вроде видеоконференцсвязи, но как правило не используется

mtu N

позволяет установит MTU для интерфейса

netmask <**addr**>

этот параметр позволяет задать маску сети, в которой находится данный интерфейс

irq <**addr**>

этот параметр работает только с определенными типами сетевых карт и позволяет задать IRQ для соответствующей интерфейсу карты

[- broadcast [addr]]

этот параметр разрешает прием широковещательных пакетов на заданный адрес либо запрещает прием таких пакетов.

[- pointopoint [addr]]

этот параметр позволяет установить адрес машины на противоположном конце соединения точка-точка (например *slip* или *ppp*)

hw <**type**> <**addr**>

этот параметр позволяет задать аппаратный адрес некоторых типов сетевых устройств. Эта опция редко используется в сетях ethernet, но очень полезна в сетях других типов сетей, таких как AX.25

Вы можете использовать программу *ifconfig* для любого сетевого интерфейса. Некоторые программы, такие как *pppd* и *dip* автоматически конфигурируют сетевой интерфейс после создания и не требуют дальнейшей ручной настройки.

5.5 Настройка Системы Преобразования Имен (Name Resolver)

Система преобразования имен — часть стандартной библиотеки Линукса. Ее основное назначение — преобразовывать понятные пользователю имена вроде '`ftp.funet.fi`' в понятные компьютеру IP-адреса, такие как `128.214.248.6`.

5.5.1 Что такое "имя"?

Вы, по-видимому, знакомы с тем, как выглядят имена машин в интернете, но можете не знать, как они конструируются. Система имен в интернет является иерархической, организованной в виде дерева. *Доменом* называется семейство имен. *Домен* может быть разбит на *поддомены*. Домены, которые не являются поддоменами других доменов, называются *доменами верхнего уровня*. Имена доменов верхнего уровня описаны в RFC920. В частности это следующие домены:

COM

Коммерческие организации

EDU

Учебные заведения

GOV

Правительственные организации

MIL

Военные организации

ORG

Другие организации

NET

Сетевые организации

Код страны

Двухбуквенный код ISO, соответствующий определенной стране

По историческим причинам большая часть доменов, не заканчивающихся на код страны относится с США, хотя Соединенные Штаты имеют свой код '.us'. В настоящее время, однако, домены .com and .org широко используются и не-американскими организациями.
Каждый из этих доменов имеет свои поддомены. Домены стран часто разделяются на поддомены com, edu, gov, mil и org. Например домены коммерческих и правительственных организаций в Австралии заканчиваются на com.au и gov.au соответственно. Заметим, однако, что это не всеобщая практика, в некоторых странах такого разбиения не производится.
Следующее слово в доменном имени обычно соответствует названию организации. Дальнейшая детализация может быть разной, например представлять структуру подразделений и отделов в организации, или быть построенной по другому принципу, выбранному сетевым администратором. Первое слово в имени — уникальное (в рамках домена) '*имя машины*'. Оставшаяся часть называется '*именем домена*', а в целом они составляют '*Полностью определенное доменное имя*'.
Например в имени 'perf.no.itg.telstra.com.au' именем машины является 'perf' а именем домена — 'no.itg.telstra.com.au'. Этот домен принадлежит к домену страны Австралия, поддомену Австралийских коммерческих организаций. Организация называется 'telstra', а внутренняя структура поддомена организации соответствует ее административной структуре — в нашем примере машина принадлежит отделу 'Network Operations' (no) в подразделении 'Information Technology Group' (itg).
Обычно имена получаются более короткими, например, интернет-провайдер "systemy.it" и организация "linux.it" без поддоменов com и org. Машина имеет имя "morgana.systemy.it", а e-mail адрес — rubini@linux.it. Владелец домена имеет право регистрировать и машины и поддомены в рамках своего домена, например, группа пользователей Линукса имеет поддомен pluto.linux.it, зарегистрированный у владельцев домена linux.it.

5.5.2 Что Вам нужно знать ?

Вы должны знать, какому домену принадлежат Ваши машины. Программы преобразования имен используют для работы '*Сервер Преобразования Имен*' (Domain Name Server, DNS), поэтому Вы должны знать адрес этого сервера.
После этого вы должны отредактировать следующие три файла:

5.5.3 /etc/resolv.conf

Главный файл конфигурации системы преобразования имен — /etc/resolv.conf. Он имеет весьма простой формат. Это текстовый файл, каждая строка которого задает один из параметров системы преобразования имен. Как правило, используются три ключевых слова-параметра:

domain

Задает имя локального домена.

search

Задает список имен доменов, которые будут добавляться к имени машины, если Вы не укажете явно имени домена для этой машины

nameserver

Этот параметр, который вы можете указывать несколько раз, задает IP-адрес сервера преобразования имен, на который ваша машина будет посылать запросы. Повторяя этот параметр, Вы можете задать несколько серверов.

Например, `/etc/resolv.conf` может выглядеть так:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

В этом примере машина находится в домене `maths.wu.edu.au`. Если Вы зададите имя машины, не указывая домена, например '`boulder`', то система преобразования имен попытается сначала найти машину '`boulder.maths.wu.edu.au`', а в случае неудачи — '`boulder.wu.edu.au`'. Для преобразования имен Ваша машина будет обращаться к серверам по адресам '`192.168.10.1`' и '`192.168.12.1`'.

5.5.4 /etc/host.conf

В файле `/etc/host.conf` задаются параметры, влияющие на поведение системы преобразования имен. Полностью формат этого файла описан на man-странице '`resolv+`', но в большинстве случаев будет достаточно такого файла:

```
order hosts,bind
multi on
```

Эти параметры указывают системе преобразования имен, что надо просмотреть файл `/etc/hosts` перед тем, как посылать запрос к серверу, и что следует возвращать все найденные в `/etc/hosts` адреса для данного имени, а не только первый.

5.5.5 /etc/hosts

В этом файле Вы можете указать имена и IP-адреса машин в локальном домене. Для преобразования имен из этого файла Вашей машине не прийдется обращаться к серверу. Недостаток такого подхода в том, что Вам надо самостоятельно следить за всеми изменениями в структуре Вашей локальной сети и вносить изменения в файл `/etc/hosts`, если IP-адрес какой-нибудь из машин изменится. Как правило в этом файле указывают только 2 имени:

```
# /etc/hosts
127.0.0.1       localhost loopback
192.168.0.1     this.host.name
```

Обратите внимание на первую строчку. Она демонстрирует, что Вы можете указать более одного имени для IP-адреса. Адрес '`127.0.0.1`' всегда используется для так называемого '`кольцевого интерфейса`'.

5.5.6 Настройка Сервера Преобразования Имен

Вы можете запустить на своей машине локальный сервер имен. Это несложно. Подробно этот процесс описан в *DNS-HOWTO* <DNS-HOWTO.html> или в любой документации по системе *BIND* (Berkeley Internet Name Domain).

5.6 Настройка кольцевого интерфейса.

'Кольцевой' интерфейс — специальный интерфейс, связывающий Вашу машину саму с собой. Его можно использовать для разных целей. Например, Вы можете захотеть протестировать какое-либо сетевое программное обеспечение, не вмешиваясь в работу других машин в сети. Для этого интерфейса специально зарезервирован адрес '127.0.0.1'. Поэтому, запустив telnet на адрес 127.0.0.1 на любой машине, получите соединение с этой же машиной. Настройка кольцевого интерфейса очень проста (Как правило эти действия выполняет один из инициализационных скриптов).

```
root# ifconfig lo 127.0.0.1
root# route add -host 127.0.0.1 lo
```

Команда *route* будет подробно рассмотрена в следующем разделе.

5.7 Маршрутизация.

Маршрутизация — широкая тема, и о ней можно можно написать очень много. Для большинства, однако, схема маршрутизации будет достаточно простой, поэтому здесь будет рассмотрены только базовые вопросы, связанные с маршрутизацией. Если Вы интересуетесь этой темой, то можете обратиться к источникам, указанным в начале этого документа.
Начнем с определения. Итак, что же такое IP-маршрутизация?

> IP-маршрутизацией называется процесс, при помощи которого машина с несколькими сетевыми интерфейсами выбирает, через какой из интерфейсов послать полученный IP-пакет.

Проиллюстрируем это определение на примере. Представьте себе обычный офисный маршрутизатор, который имеет PPP-соединение с интернетом, несколько подключенных сегментов локальной ethernet-сети и PPP-соединение с другим офисом. Когда через один из интерфейсов на маршрутизатор приходит IP-пакет, маршрутизатор должен решить, через какой из интерфейсов отправлять этот пакет далее. Даже простым машинам может потребоваться маршрутизировать пакеты, так как они имеют по меньшей мере два интерфейса — один кольцевой, описанный выше, и второй, через который они подключены к реальной сети, например ethernet-интерфейс, или интерфейс подключения по последовательным линиям PPP или SLIP.
Как же происходит маршрутизация? На каждой машине хранится специальный список правил маршрутизации, называемый таблицей маршрутизации. Любая строка этой таблицы как правило содержит по меньшей мере три поля. Первое поле — адрес назначения, второе — имя интерфейса, через который следует отправлять пакеты, и третье — IP-адрес машины, через которую будут передаваться пакеты. В Линуксе, Вы можете просмотреть таблицу маршрутизации с помощью следующей команды:

```
user% cat /proc/net/route
```

или одной из команд:

```
user% /sbin/route -n
user% netstat -r
```

Процесс маршрутизации достаточно прост: когда машина получает IP-пакет, его адрес назначения (адрес машины, на которую отправлен этот пакет) сверяется с каждой строкой таблицы маршрутизации. Выбирается подходящая строка и пакет передается через указанный в этой строке интерфейс. Если поле адреса 'машины-передатчика' заполнено, то пакет передается на эту машину, иначе считается, что адресат пакета находится в сети, к которой подключен данный интерфейс.

Для управления таблицей маршрутизации используется программа '*route*'. Эта программа преобразует свои аргументы в параметры вызова ядра, и ядро добавляет, удаляет или изменяет строки в таблице маршрутизации.
Простой пример. Представьте себе, что у вас есть ethernet-сеть. Она организована как сеть класса C с адресом '192.168.1.0'. Вашей машине был выделен адрес '192.168.1.10' и было сказано, что маршрутизатор, через который Ваша сеть подключена к интернету находится по адресу '192.168.1.1'.
Первым делом Вы должны настроить сетевой интерфейс. Команда будет выглядеть так:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

После этого Вы должны добавить в таблицу маршрутизации на Вашей машине строку, согласно которой пакеты на машины с адресами '192.168.1.*' ядро должно отправлять через интерфейс eth0. Это делается с помощью следующей команды:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Обратите внимание на использование опции '-net'. Эта опция указывает, что адрес назначения в таблице маршрутизации будет адресом сети. С помощью опции '-host' вы можете задать маршрут на конкретный IP-адрес.
Этот маршрут позволит вам устанавливать IP-соединения со всеми машинами в вашем локальном ethernet-сегменте. Но как быть со всеми остальными машинами?
Было бы очень сложно задавать маршруты для всех возможных IP-сетей явно, поэтому используют следующий трюк — маршрут по умолчанию. Маршрут по умолчанию подходит для всех адресов назначения, не указанных в таблице маршрутизации. С помощью маршрута по умолчанию Вы говорите ядру — "а все остальное отправляй туда". В нашем примере маршрут по умолчанию настраивается командой:

```
root# route add default gw 192.168.1.1 eth0
```

Опция 'gw' указывает программе route что следующий аргумент — IP-адрес или имя маршрутизатора, на который надо отправлять все пакеты, соответствующие этой строке таблицы маршрутизации.
Итак полностью настройка будет выглядеть так:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add default gw 192.168.1.1 eth0
```

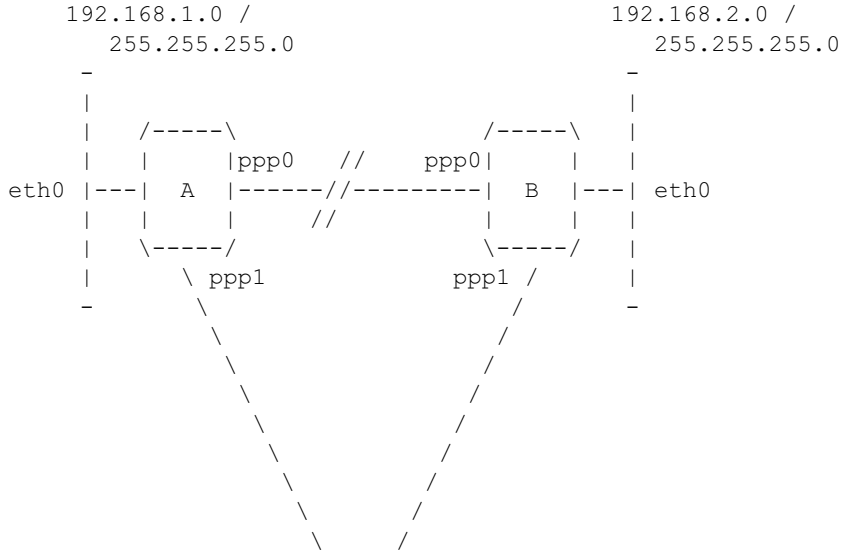
Если Вы внимательно просмотрите ваши 'rc' файлы, настраивающие сеть, Вы обнаружите, что по крайней мере один из них выглядит примерно так-же, как и в нашем примере. Приведенная конфигурация — одна из самых распространенных.
Рассмотрим теперь несколько более сложную конфигурацию маршрутизации в сети. Представьте себе, что вы должны настроить маршрутизатор из предыдущего примера. У этого маршрутизатора есть одно PPP-соединение и три подключенных ethernet-сегмента. Настройка маршрутизации будет выглядеть так:

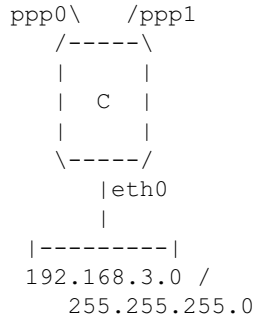
```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
root# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
root# route add default ppp0
```

Во всей сети только маршрутизатор должен иметь в своей таблице маршрутизации отдельные строки для каждой из подсетей. На остальных машинах связь с другими сегментами локальной сети будет осуществляться с помощью маршрута по умолчанию. Они будут отправлять пакеты на маршрутизатор, о тот будет передавать их в нужный сегмент сети. Вас может удивить, что маршрут по умолчанию на маршрутизаторе не использует опции 'gw'. Причина проста. Протоколы соединения по последовательным линиям, такие как PPP и SLIP всегда имеют только две машины в сети — (соединение точка-точка) поэтому указание адреса бессмысленно — на том конце соединения только одна машина. Таким образом, для подобных соединений нет нужды указывать адрес маршрутизатора, на который надо передавать пакеты. Для других типов сетей, таких как ethernet, arcnet или token ring требуется указывать адрес маршрутизатора, так как эти сети поддерживают подключение сразу многих машин к одному сегменту сети.

5.7.1 Что делает программа *routed* ?

Настройка маршрутизации в приведенных примерах лучше всего подходит для простых сетей, в которых между точками отправки и назначения существует единственный маршрут. В более сложных сетях это не так, и маршрутизация становится сложнее. К счастью, большинству из вас не прийдется сталкиваться с таким сетями.
Проблема со статической маршрутизацией, которую мы описали в примерах состоит в том, что если сетевое соединение, через которое ваша машина отправляла пакеты откажет, то единственным выходом для Вас будет вручную изменить таблицу маршрутизации. Это медленно, непрактично и чревато ошибками. Поэтому были разработаны методы, позволяющие автоматически изменять таблицы маршрутизации в случае сбоев в сети. Эти методы работают только в сетях, где между машиной-отправителем и машиной адресатом есть несколько маршрутов. Все эти методы обычно называют "Протоколами динамической маршрутизации".
Вы возможно слышали о некоторых распространенных протоколах динамической маршрутизации. Самые широко используемые протоколы — RIP (Routing Information Protocol) (Протокол информации о маршрутизации) и OSPF (Open Shortest Path First Protocol) (Протокол кратчайшего открытого пути). Протокол RIP используется в основном в небольших сетях, таких как сеть небольшой или средней организации. Более современный протокол OSPF обладает бОльшими возможностями для управления большими сетями в которых существует много маршрутов между отправителем и адресатом. Самые распространенные реализации этих протоколов — программы 'routed' — RIP и 'gated' — RIP, OSPF и другие. Программа 'routed' обычно включается в дистрибутивы Линукса и включена в пакет 'NetKit', упомянутый выше.
Пример использования динамической маршрутизации может выглядеть примерно так:





У Вас есть три маршрутизатора — A, B и C. Каждый из них обслуживает по одному ethernet-сегменту с IP-сетью класса C (маска 255.255.255.0). Кроме того, каждый из них имеет PPP-соединения с двумя другими. Сеть имеет форму треугольника.
Таблица маршрутизации на маршрутизаторе A задается следующими командами:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
root# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Все будет нормально, пока не произойдет сбой на соединении между маршрутизаторами A и B. В случае сбоя машины из сегмента A не смогут установить соединения с машинами из сегмента B, так как маршрутизатор A будет пытаться передать пакеты через интерфейс ppp0. В то же время он смогут связываться с машинами из сегмента C, а машины из сегмента C смогут связываться с машинами из сегмента B.
Но раз сегмент A "видит"сегмент B, а тот, в свою очередь "видит"сегмент C, то почему бы маршрутизатору A не передавать пакеты для машин из сегмента B через маршрутизатор C ? Именно это и помогает осуществить динамическая маршрутизация. Если на каждом из маршрутизаторов будет запущена демон динамической маршрутизации, то их таблицы маршрутизации будут автоматически изменены, чтобы подстроиться под изменения, произошедшие в структуре сети при сбое одного из соединений. Для настройки динамической маршрутизации в нашем примере будет достаточно выполнить на каждом из маршрутизаторов по две команды. Например, на маршрутизаторе A:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# /usr/sbin/routed
```

Демон '`routed`' при запуске автоматически найдет все активные сетевые интерфейсы и начнет передавать и принимать специальные пакеты, с помощью которых он будет постоянно иметь представление о текущей структуре сети, и настраивать в соответствии с ней таблицу маршрутизации. Это очень краткое описание динамической маршрутизации и сетей, в которых Вам следует ее использовать. Для получения более подробной информации обратитесь к ссылкам, приведенным в начале этого документа.
Резюмируя вышесказанное:

1. Динамическая маршрутизация нужна лишь в тех случаях, когда на Вашей машине есть выбор между маршрутами, по которым вы можете отправлять пакеты.
2. Демон динамической маршрутизации будет автоматически изменять таблицу маршрутизации на Вашей машине при любых изменениях в структуре сети.
3. Протокол RIP предназначен для небольших и средних сетей.

5.8 Настройка сетевых серверов и сервисов.

Сетевые сервера и серверы — это программы, позволяющие пользователям на других машинах использовать Вашу машину. Сетевые серверы "слушают"сетевые порты. Сетевые порты — средство вызова нужного сервиса на нужной машине. С их помощью машина отличает входящие telnet-соединения от входящих ftp-соединений. Удаленный пользователь запрашивает сетевое соединение с вашей машиной, а сервер на вашей машине, "слушающий"запрашиваемый порт устанавливает и обслуживает соединение. Сервера обычно работают в одном из двух режимов:

самостоятельный

сетевой сервер (демон) слушает один из сетевых портов, и при запросе входящего соединения открывает и обслуживает его.

работающий под управлением демона *inetd*

демон *inetd* — особый сетевой сервер, занимающийся управлением входящими сетевыми соединениями. В его файле конфигурации описано, какая программа должна быть запущена при запросе на входящее соединение. Каждый из портов в этом файле описывается как обслуживающий соединения по протоколу tcp или протоколу udp, или обоим этим протоколам. Порты описываются в другом файле, который будет вскоре рассмотрен.

Вам нужно настроить два важных файла — `/etc/services`, в котором числовым номерам портов присваиваются мнемонические имена и `/etc/inetd.conf` — конфигурационный файл для демона *inetd*.

5.8.1 /etc/services

Файл `/etc/services` — простая таблица, задающая соответствия между числовыми номерами портов и читаемыми мнемоническими именами. Формат этого файла очень прост. Это текстовый файл с тремя полями в каждой строчке:

```
имя             порт/протокол   псевдонимы      # комментарий
```

имя

слово, задающее имя сервиса.

порт/протокол

это поле разбито на два подполя.

порт

число, задающее номер порта, на котором находится данный сервис. Большинство основных сервисов имеют зарезервированные номера портов. Эти номера описаны в `RFC1340`.

протокол

подполе, содержащее `tcp` или `udp`

Очень важно понимать, что строки, содержащие поля `10/tcp` и `10/udp` вообще говоря не связаны между собой, поэтому сервис, привязанный к `10/tcp` совсем не обязан быть привязанным к `10/udp`. Как правило, обе строки в таблице присутствуют только для тех сервисов, которые реально обслуживают соединения по обоим протоколам.

псевдонимы

другие имена, которые можно использовать для задания данного сервиса.

Любой текст после символа '#' считается комментарием и игнорируется.

Пример файла `/etc/services`. Все современные дистрибутивы Линукса содержат весьма полный файл '`/etc/services`'. На случай, если Вы устанавливаете систему не из дистрибутива, приводим файл `/etc/services`, включенный в дистрибутив *Debian* <http://www.debian.org/>.

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, ``Assigned Numbers'' (July 1992).  Not all ports
# are included, only the more common ones.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
```

```
www             80/udp                          # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5  # Kerberos v5
kerberos        88/udp          kerberos5 krb5  # Kerberos v5
supdup          95/tcp
# 100 - reserved
hostnames       101/tcp         hostname        # usually from sri-nic
iso-tsap        102/tcp         tsap            # part of ISODE.
csnet-ns        105/tcp         cso-ns          # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                         # Remote Telnet
rtelnet         107/udp
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
```

```
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized `rlogin' (v5)
kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# From ``Assigned Numbers'':
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical
#> connections which carry long term conversations.  For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined.  This list specifies the port used by the server process as its
#> contact port.  While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convienence to the
#> community.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually uses UDP only
bbs             7000/tcp                # BBS service
#
```

```
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4 and are unofficial.  Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Local services
```

В реальности этот файл постоянно растет по мере появления новых сервисов. Если Вы опасаетесь, что Ваш экземпляр этого файла недостаточно полон, возьмите его из одного из последних дистрибутивов.

5.8.2 /etc/inetd.conf

Файл `/etc/inetd.conf` — файл конфигурации для демона *inetd*. Его назначение в том, чтобы описать, какие действия должен выполнить *inetd* при получении входящего запроса. Для каждого из сервисов, обслуживаемых демоном *inetd* Вы должны указать, какую программу надо запустить для обслуживания этого соединения и как ее запускать.
Формат этого файла также весьма прост. Это текстовый файл, в каждой строке которого описывается один из сервисов. Любой текст после символа '#' до конца строки считается комментарием и игнорируется. Каждая строка содержит семь полей, разделенных произвольным количеством пробелов или символов табуляции. Формат строки таков:

```
сервис тип_сокета протокол флаг имя_пользователя путь_к_серверу параметры_сервера
```

сервис

имя сервиса из файла `/etc/services`

тип_сокета

задает какого типа сокет следует создавать для обслуживания соединения. Допустимыми значениями являются `stream`, `dgram`, `raw`, `rdm`, `seqpacket`. Не вдаваясь в технические подробности можете пользоваться следующим правилом — как правило соединения по протоколу `tcp` используют тип `stream`, а соединения по протоколу `udp` — тип `dgram`. Только для нескольких специальных сетевых серверов используются другие значения этого поля.

протокол

протокол, считающийся допустимым для данного сервиса. Это должен быть один из протоколов, описанных в файле `/etc/services` для данного сервиса, и как правило имеет значение `tcp` или `udp`. Серверы на базе протокола "Вызовов удаленных процедур"(RPC, Remote Procedure Call) фирмы Sun имеют в этом поле значения `rpc/tcp` или `rpc/udp`.

флаг

это поле может иметь одно из двух значений — `wait` и `nowait`. В зависимости от значения этого поля, демон *inetd* будет запускать несколько экземпляров сервера, обслуживающего данное соединение, или будет дожидаться завершения работы сервера, предполагая, что сервер самостоятельно обслужит все приходящие во время его работы запросы. Как правило, для `tcp`-сервисов это поле имеет значение `nowait`, а для `udp`-сервисов — `wait`. Впрочем из этого правила есть несколько важных исключений, так что будьте внимательны и обратитесь к документации сервера, если не уверены.

имя_пользователя

задает, с какими полномочиями будет работать запущенный сервер. это имя из файла `/etc/passwd` используется из соображений безопасности. Установив его в `nobody` Вы минимизируете риск и возможный ущерб при "взломе"вашего сервера. Впрочем, многие из важных серверов требуют привилегий пользователя "root"для того, чтобы правильно функционировать.

путь_к_серверу

путь для запуска программы-сервера

параметры_сервера

это поле может быть опущено. В нем вы можете указать, какие параметры следует передать в командной строке сервера при его запуске.

Пример файла `/etc/inetd.conf` Как и в случае с файлом `/etc/services`, большинство современных дистрибутивов содержат достаточно полный файл `/etc/inetd.conf` Приведем в качестве примера файл `/etc/inetd.conf` из дистрибутива *Debian* <http://www.debian.org/>.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emden.de>
#
```

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Internal services
#
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
daytime         dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# These are standard services.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et al
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# `cfinger' is for the GNU finger server available for Debian.  (NOTE: The
# current implementation of the `finger' daemon allows it to be run as `root'.)
#
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
#
# Kerberos authenticated services (these probably need to be corrected)
#
```

```
#klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be corrected)
#
#krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# RPC based services
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident           stream  tcp     nowait  nobody  /usr/sbin/identd        identd -i
```

5.9 Другие файлы, связанные с настройкой сети.

Информация о настройке сети содержится еще в некотором количестве файлов. Вам по-видимому не потребуется изменять их, но в любом случае полезно представлять себе, как они устроены и для чего предназначены.

5.9.1 /etc/protocols

Файл `/etc/protocols` ставит в соответствие номера и имена протоколов. Он позволяет программистам указывать в своих программах протоколы, пользуясь легко запоминаемыми мнемоническими именами, а также некоторыми программами, такими как *tcpdump*, для отображения мнемонических имен протоколов. Формат этого файла таков:

```
имя_протокола  номер  псевдонимы
```

Файл `/etc/protocols` из дистрибутива *Debian* <http://www.debian.org/>

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
```

```
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2 /etc/networks

Файл `/etc/networks` имеет почти то же самое значение, что и файл `/etc/hosts`. Он позволяет задавать имена для сетевых IP-адресов. В отличие от `/etc/hosts`, в каждой строке этого файла задается два поля:

```
имя_сети  адрес_сети
```

Например:

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

При запуске программы *route*, если адрес назначения является сетевым, Вы можете использовать имя из файла `/etc/networks` вместо IP-адреса.

5.10 Безопасность и управление доступом.

Хотелось бы начать этот раздел с предупреждения, что обеспечение безопасности Вашей машины — сложная вещь. Здесь описываются только самые простые механизмы защиты машины в сети от несанкционированного доступа. Если Вы желаете заняться этим вопросом более серьезно, Вам прийдется ознакомится самостоятельно с информацией на эту тему, доступной в интернете, в том числе *Security-HOWTO* <Security-HOWTO.html>
Важное и простое правило обеспечения безопасности — '**Не запускайте ненужные сервера.**"Многие дистрибутивы устроены таким образом, что стартуют после запуска все установленные сервера. Для обеспечения минимального уровня безопасности просмотрите файл `/etc/inetd.conf` и закомментарьте (поставьте символ '#' в начале строки) во всех строчках с сервисами, которые Вы не собираетесь использовать. Хорошими кандидатами для этого будут такие сервисы как `shell`, `login`, `exec`, `uucp`, `ftp` и информационные сервисы — `finger`, `netstat`, `systat`.
Из всего многообразия механизмов обеспечения безопасности здесь будут описаны самые простые.

5.10.1 /etc/ftpusers

Файл `/etc/ftpusers` — простейший механизм, позволяющий Вам запретить некоторым пользователям получать доступ к Вашей машине по ftp. Этот файл используется программой-сервером ftp (*ftpd*) при обработке входящего соединения. Этот файл содержит список имен пользователей, которые не имеют права работать с вашей машиной по ftp. Он может выглядеть например так:

```
# /etc/ftpusers - пользователи, которым запрещен доступ по ftp
root
uucp
bin
mail
```

5.10.2 /etc/securetty

Файл /etc/securetty задает список устройств tty, с которых в систему может входить пользователь 'root'. Этот файл используется программой регистрации в системе (обычно /bin/login). Он содержит список устройств, которыми можно пользоваться для работы в системе под именем root, на всех остальных устройствах пользователь root не сможет войти в систему.

```
# /etc/securetty - tty, с которых может работать пользователь root
tty1
tty2
tty3
tty4
```

5.10.3 Механизм управления доступом с помощью программы *tcpd*

Вы уже встречали упоминание о программе *tcpd* в файле /etc/inetd.conf. Эта программа обеспечивает протоколирование и ограничение доступа к тем сервисам, которые она защищает. При запуске из демона *inetd* она читает два файла, содержащие правила разрешения или запрещения доступа, и действует в соответствии с этими правилами. Эта программа будет искать файлы /etc/hosts.allow и /etc/hosts.deny. Если оба эти файла отсутствуют, то доступ к любому из сервисов разрешается. Структура этих файлов будет описана далее, а за подробностями о работе программы *tcpd* обратитесь к соответствующим man-страницам (советуем Вам начать со страницы hosts_access(5)).

/etc/hosts.allow Файл /etc/hosts.allow — один из конфигурационных файлов программы */usr/sbin/tcpd*. Этот файл содержит правила, описывающие машины, с которых разрешен доступ к сервисам на Вашей машине.
Формат файла /etc/hosts.allow весьма прост:

```
# /etc/hosts.allow
#
# <список_сервисов>: <список_машин> [: команда]
```

список_сервисов

список имен серверов, разделенных запятыми, к которым относится данное правило. Например: ftpd, telnetd, fingerd.

список_машин

список имен машин, разделенных запятыми. Можете вместо имен использовать IP-адреса. Вы можете задавать группы имен. Например: gw.vk2ktj.ampr.org — описывает конкретную машину, .uts.edu.au — описывает все машины, чьи имена оканчиваются на .uts.edu.au, 44. — описывают все машины, чьи IP-адреса начинаются на 44. Есть несколько специальных значений для этого поля — например ALL обозначает любую машину, LOCAL — любую машину без символов '.' в имени (машины из Вашего домена), PARANOID — машину, чье имя

не соответствует ее адресу (возможно в результате подделки имени). Еще одно полезное значение — EXCEPT — позволяет Вам указывать исключения из списка. Все эти значения будут проиллюстрированы на примере.

команда

это поле можно опустить. оно задает полной путь к программе, которую следует запустить, когда текущее правило подходит к обрабатываемому запросу. Это может быть программа, которая будет пытаться определить, кто пытается получить доступ, или отправит почтовое сообщение или другое предупреждение системному администратору о том, что кто-то пытается получить доступ. При запуске можно использовать переменные — например переменная %h содержит имя (или адрес, если у машины нет имени) машины, которая пытается получить доступ, %d содержит имя запрашиваемого сервера.

Пример файла /etc/hosts.allow

```
# /etc/hosts.allow
#
# Разрешить всем доступ к почтовому серверу
in.smtpd: ALL
# Доступ по telnet и ftp - только из локального домена и с домашней машины
telnetd, ftpd: LOCAL, myhost.athome.org.au
# Разрешить всем доступ finger, но сообщать об этом.
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

/etc/hosts.deny Файл /etc/hosts.deny аналогичен файлу /etc/hosts.allow и содержит правила, в соответствии с которыми программа *tcpd* отказывает в соединении некоторым машинам.
Пример файла /etc/hosts.deny

```
# /etc/hosts.deny
#
# Запретить доступ всем машинам с подозрительными именами
ALL: PARANOID
#
# Запретить доступ всем
ALL: ALL
```

Строка ALL: PARANOID в данном примере лишняя, так как поглощается строкой ALL: ALL. Любая из этих строк вполне подойдет для конфигурации Вашей машины по умолчанию.
Самая безопасная конфигурация — указать правило ALL: ALL в файле /etc/hosts.deny и открыть доступ со всех нужных машин в файле /etc/hosts.allow.

5.10.4 /etc/hosts.equiv

Файл /etc/hosts.equiv используется для того, чтобы дать определенным пользователям с определенных машин доступ к вашей машине без пароля. Это может быть полезным в безопасной закрытой системе, в которой Вы контролируете все машины, в противном случае — это очень опасный механизм с точки зрения безопасности. Ваша машина будет настолько же в безопасности, насколько в безопасности наименее защищенная из машин, перечисленных в файле /etc/hosts.equiv. Из соображений безопасности не используйте этот механизм, и кроме того, советуем Вам и всем пользователям Вашей машины отказаться от использования файла .rhosts

5.10.5 Правильная настройка демона *ftp*.

На многих сайтах есть необходимость в запуске анонимного ftp-сервера, который позволяет другим получать и загружать файлы без задания имени пользователя. Если Вы планируете обеспечить такой режим доступа к Вашей машине, убедитесь что вы правильно настроили демона *ftp* на анонимный доступ. man-страницы демона *ftpd* описывают такую настройку. Убедитесь, что Вы в точности следовали этим инструкциям. Важный момент, например, состоит в том, что не следует копировать файл `/etc/passwd` в директорию `/etc`, принадлежащую анонимному пользователю, убедитесь в том что вы убрали из этого файла все строки, кроме необходимых, иначе Вы подвергаете себя риску атаки с подбором пароля.

5.10.6 Сетевые файрволлы (брандмауэры).

Отличным средством обеспечения безопасности является запрет передачи некоторых типов пакетов с помощью файрволла, так что они не достигнут Вашей машины. Эта тема подробно рассмотрена в *Firewall-HOWTO* <Firewall-HOWTO.html>, и (в общих чертах) далее в этом документе.

5.10.7 Другие соображения.

Предложим несколько других соображений (почти религиозного плана) на Ваше рассмотрение:

Программа sendmail

несмотря на популярность программы *sendmail*, постоянно появляются сообщения о проблемах с безопасностью этой программы.

Сетевая файловая система NFS и другие сервисы, основанные на протоколе RPC:

Эти сервисы подвержены большому количеству возможных атак. Им трудно найти замену, и если уж Вы установили их, убедитесь в правильности назначения прав доступа.

6 Информация об IP- Ethernet-сетях.

В этом разделе рассматриваются вопросы работы ethernet и IP сетей. Фактически здесь собраны наиболее интересные разделы из относящихся к конкретным сетевым технологиям, и они будут полезны всем тем, что использует Линукс в локальных сетях.

6.1 Ethernet

Ядро присваивает ethernet-устройствам имена‘`eth0`’, ‘`eth1`’, ‘`eth2`’ и т.д. Первая обнаруженная карта получает имя ‘`eth0`’, а все остальные нумеруются по порядку обнаружения.
По умолчанию ядро пытается обнаружить только одно ethernet-устройство, если у Вас в машине несколько ethernet-карт, то Вам потребуется указать в командной строке запуска ядра параметры для обнаружения оставшихся карт.
Подробно работа ethernet-карт под Линуксом описана в *Ethernet-HOWTO* <Ethernet-HOWTO.html>.
После того как ядро будет скомпилировано с поддержкой Вашей ethernet-карты, Вам достаточно выполнить подобные следующим команды для ее настойки:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

Большинство драйверов ethernet-карт было написано Дональдом Беккером (Donald Becker, `becker@CESDIS.gsfc.nasa.gov`).

6.2 EQL – балансировщик потока данных.

Это устройство имеет имя 'eql' (оно может быть только одно) и служит для объединения нескольких соединений точка-точка, таких как PPP, slip или plip в одно соединение, по которому передаются IP-пакеты. Иногда оказывается дешевле использовать несколько низкоскоростных соединений, чем одно высокоскоростное.
Опции компиляции ядра:

```
Network device support  --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

Для работы такого соединения необходимо, чтобы машина на другой стороне также поддерживала eql. Сейчас такая поддержка есть в Линуксе, Livingstone Portmasters и некоторых современных dial-in серверах.
Для настройки EQL вам понадобятся утилиты поддержки eql, которые можно получит по адресу *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz>.
Настройка достаточно проста. Первым делом необходимо настроить eql-интерфейс. Он настраивается так же, как и любое другое сетевое устройство. IP-адрес и mtu настраиваются программой *ifconfig*:

```
root# ifconfig eql 192.168.10.1 mtu 1006
```

После этого Вы должны настроить все Ваши реальные соединения точка-точка. Способ настройки зависит от типа соединения – обратитесь к соответствующему разделу этого документа за подробностями.
И наконец Вы должны связать все эти соединения с eql. Этот процесс называют 'подчинением' и выполняется с помощью программы *eql_enslave*:

```
root# eql_enslave eql sl0 28800
root# eql_enslave eql ppp0 14400
```

Параметр '*ожидаемая скорость соединения*' (последний параметр в примере) оказывает косвенное влияние на работу eql. Он определяет долю пакетов, передаваемых через соответствующее соединение, и Вы можете пытаться повысить производительность eql, меняя этот параметр.
Для отсоединения сетевого интерфейса от eql используйте программу *eql_emancipate*:

```
root# eql_emancipate eql sl0
```

При настройке маршрутизации замените в командах *route* все 'подчиненные' интерфейсы на eql. Обычно это выглядит так:

```
root# route add default eql
```

Драйвер EQL был написан Саймоном Джейнсом (Simon Janes, simon@ncm.com).

6.3 IP-учет (для версий ядра 2.0).

Функция IP-учета позволяет ядру собирать и анализировать информацию об использовании сети. Ядро собирает данные о количестве пакетов и количестве байт, переданных по сети с момента последнего сброса этих данных. Вы можете задать различные правила для того, чтобы классифицировать эти данные. В ядре версии 2.1.102 эта возможность была временно изъята, так как старая

программа настройки файрволла ipfwadm, которая используется и для настройки IP-учета, была заменена на "ipfwchains".

Опции компиляции ядра:

```
Networking options  --->
    [*] IP: accounting
```

После того, как Вы откомпилировали и установили ядро с поддержкой IP-учета, используйте программу *ipfwadm* для его настройки. Вам может потребоваться разбивать учетную информацию по многим признакам. Ниже приведен простой, но достаточно полезный пример, за более детальной информацией обратитесь к man-странице программы *ipfwadm*.

Сценарий: У Вас есть ethernet-сеть, подключенная к интернету через PPP-соединение. На одной из машин в сети запущено большое количество сервисов и Вы хотели бы знать какой объем данных передается сервисами telnet, rlogin, ftp и http.

Вы можете использовать следующий скрипт:

```
#!/bin/sh
#
# Сброс правил учета
ipfwadm -A -f
#
# Правила для локальной сети
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
ipfwadm -A in -a -P tcp -D 44.136.8.96/29
ipfwadm -A out -a -P tcp -D 44.136.8.96/29
ipfwadm -A in -a -P udp -D 44.136.8.96/29
ipfwadm -A out -a -P udp  -D 44.136.8.96/29
ipfwadm -A in -a -P icmp -D 44.136.8.96/29
ipfwadm -A out -a -P icmp -D 44.136.8.96/29
#
# Правила по умолчанию
ipfwadm -A in -a -P tcp -D 0/0 20
ipfwadm -A out -a -P tcp -S 0/0 20
ipfwadm -A in -a -P tcp -D 0/0 23
ipfwadm -A out -a -P tcp -S 0/0 23
ipfwadm -A in -a -P tcp -D 0/0 80
ipfwadm -A out -a -P tcp -S 0/0 80
ipfwadm -A in -a -P tcp -D 0/0 513
ipfwadm -A out -a -P tcp -S 0/0 513
ipfwadm -A in -a -P tcp -D 0/0
ipfwadm -A out -a -P tcp -D 0/0
ipfwadm -A in -a -P udp -D 0/0
ipfwadm -A out -a -P udp  -D 0/0
ipfwadm -A in -a -P icmp -D 0/0
ipfwadm -A out -a -P icmp -D 0/0
#
# Распечатать список правил
ipfwadm -A -l -n
#
```

Имена “ftp-data" и “www" — имена сервисов из файла `/etc/services`. Последняя команда печатает список правил и накопленные данные.
Следует обратить внимание на то, что при обработке пакета **величины накопленных данных во всех подходящих правилах будут увеличены**, поэтому Вам потребуется произвести некоторые вычисления для того, чтобы получить интересующие Вас данные. Например, для того, чтобы узнать какое количество данных было передано "мимо"telnet, rlogin, ftp или http, необходимо вычесть из их данные из данных правила, которое описывает все порты.

```
root# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source              destination         ports
    0     0 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> 20
    0     0 out tcp  44.136.8.96/29      0.0.0.0/0           20 -> *
   10  1166 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> 80
   10   572 out tcp  44.136.8.96/29      0.0.0.0/0           80 -> *
  252 10943 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> *
  231 18831 out tcp  44.136.8.96/29      0.0.0.0/0            * -> *
    0     0 in  udp  0.0.0.0/0           44.136.8.96/29      * -> *
    0     0 out udp  44.136.8.96/29      0.0.0.0/0           * -> *
    0     0 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> 20
    0     0 out tcp  0.0.0.0/0           0.0.0.0/0           20 -> *
   10  1166 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> 80
   10   572 out tcp  0.0.0.0/0           0.0.0.0/0           80 -> *
  253 10983 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> *
  231 18831 out tcp  0.0.0.0/0           0.0.0.0/0           * -> *
    0     0 in  udp  0.0.0.0/0           0.0.0.0/0           * -> *
    0     0 out udp  0.0.0.0/0           0.0.0.0/0           * -> *
```

6.4 IP-учет (для версий ядра 2.2)

Новая система учета использует систему “IP Firewall Chains". Обратитесь к *странице системы IP-цепочек* <http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html> за более детальной информацией. Среди прочего, Вы будете должны использовать программу *ipchains* вместо программы `ipfwadm` для настройки IP-учета. (Информация взята из файла `Documentation/Changes` последней версии исходных текстов ядра).

6.5 IP-псевдонимы.

Иногда оказывается полезным, чтобы одному сетевому устройству соответствовало несколько IP-адресов. Например, эта функция используется интернет-провайдерами для создания www или ftp-сайтов своих клиентов. Несколько более подробное описание IP-псевдонимов дается в “IP-Alias mini-HOWTO".
Опции компиляции ядра:

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

После того, как вы откомпилируете и установите ядро с поддержкой IP-псевдонимов, дальнейшие настройки достаточно просты. IP-псевдонимы присваиваются виртуальным устройствам, связанным с реальным устройством. Имена этим устройствам присваиваются по правилу `<имя_устройсва>:<номер_виртуального_устройства>`, например `eth0:0` или `ppp0:10`. Такое устройство нужно конфигурировать *после* настройки основного интерфейса.

Предположим, что у Вас есть ethernet-сеть с двумя существующими одновременно IP-сетями, и Вы хотите, чтобы Ваша машина имела доступ к обеим этим сетям. Для этого выполните следующие команды:

```
root# ifconfig eth0:0 192.168.1.1 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0:0

root# ifconfig eth0:1 192.168.10.1 netmask 255.255.255.0 up
root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Для удаления псевдонима просто добавьте символ '-' к имени устройства:

```
root# ifconfig eth0:0- 0
```

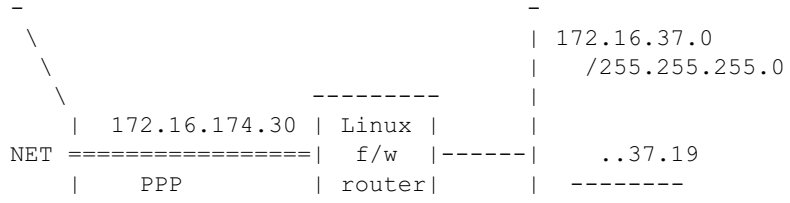
Все данные о маршрутизации через этот псевдоним будут автоматически удалены.

6.6 IP файрволл (для версий ядра 2.0).

Использование файрволлов подробно рассмотрено в *Firewall-HOWTO* <Firewall-HOWTO.html>. IP-файрволл позволяет вам предотвращать несанкционированный доступ к Вашей машине путем отбрасывания IP-пакетов по заданным правилам. Есть три типа правил — входные фильтры, выходные фильтры и фильтры передачи. Входные фильтры применяются к пакетам, приходящим из сети. Выходные фильтры применяются к пакетам, предназначенным к отправке в сеть. Фильтры передачи применяются к полученным пакетам, которые не предназначены для данной машины и должны быть маршрутизированы.
Опции компиляции ядра:

```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

Задание фильтров производится с помощью программы *ipfwadm*. Данный документ не ставит своей задачей углубляться в тонкости сетевой безопасности, поэтому, если Вы хотите надежно защитить свою сеть, вам потребуется самостоятельно ознакомится с этим вопросом.
По-видимому наиболее распространенный случай использования IP-файрволла — это когда Ваша машина является маршрутизатором, через который локальная сеть подключена в интернету, и Вы хотите предотвратить несанкционированный доступ к машинам Вашей локальной сети с машин из внешних сетей.
Данный пример был любезно предоставлен Арнтом Гюлбрандсеном (Arnt Gulbrandsen, <agulbra@troll.no>).
Данный пример иллюстрирует настройку фильтров для маршрутизатора, изображенного на этом рисунке:



Приведенные ниже команды настройки файрволла следует поместить в один из `rc`-файлов, так чтобы они автоматически выполнялись при запуске системы. Для обеспечения максимальной безопасности их следует выполнять после настройки сетевых интерфейсов, но до их активизации, чтобы предотвратить возможность несанкционированного доступа в момент загрузки системы.

```
#!/bin/sh

# Сбросить таблицу фильтров передачи
# Установить правило по умолчанию в 'разрешить'
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# То же самое для входных фильтров
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# Настроить интерфейс PPP
# Можно было бы использовать опцию '-a deny' вместо '-a reject -y'
# но тогда будет невозможно открывать исходящие соединения на этом
# интерфейсе. Опция '-o' указывает, что отвергнутые пакеты следует
# протоколировать. Тратя место на диске, вы получаете возможность
# обнаруживать атаки и ошибки в конфигурации.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Отбрасывать очевидно неверные пакеты:
# Информация не должна приходит с любых типов широковещательных адресов
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# Пакеты с кольцевого интерфейса не должны попадать на реальный
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# разрешить входящие SMTP и DNS запросы, но только к выделенному для
# этого серверу
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS использует протокол UDP наряду с TCP, его тоже следует разрешить
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# запретить "ответы" на опасные порты, такие как NFS или его расширений
# (Larry McVoy's NFS extension). Если у Вас работает squid, добавьте и
# его порты
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050

# ответы на другие порты разрешены
```

```
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Запретить входящие соединения с демоном identd
# Используйте параметр 'reject' чтобы машина, пытающаяся установить
# соединение получала отказ немедленно
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Разрешить соединения определенных типов из "дружественных" сетей
# 192.168.64 и 192.168.65.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23

# Разрешить прохождение любых пакетов из локальной сети.
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# запретить остальные tcp-соединения и протоколировать их
# (добавьте 1:1023 если у Вас перестанет работать ftp)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# то же самое для udp-соединений
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Правильная настройка файрволла — нелегкая задача. Приведенный пример может послужить хорошей отправной точкой. Некоторую информацию Вы можете получить, воспользовавшись man-страницей программы *ipfwadm*. Обязательно получите информацию из всех возможных надежных источников и попросите кого-либо протестировать ваши настройки "снаружи".

6.7 IP-файрволл (для версий ядра 2.2)

Новый файрволл использует систему "IP Firewall Chains". Обратитесь к *странице системы IP-цепочек* <http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html> за более детальной информацией. Среди прочего, Вы будете должны использовать программу *ipchains* вместо программы `ipfwadm` для настройки IP-файрволла. (Информация взята из файла `Documentation/Changes` последней версии исходных текстов ядра).

6.8 IPIP-включение

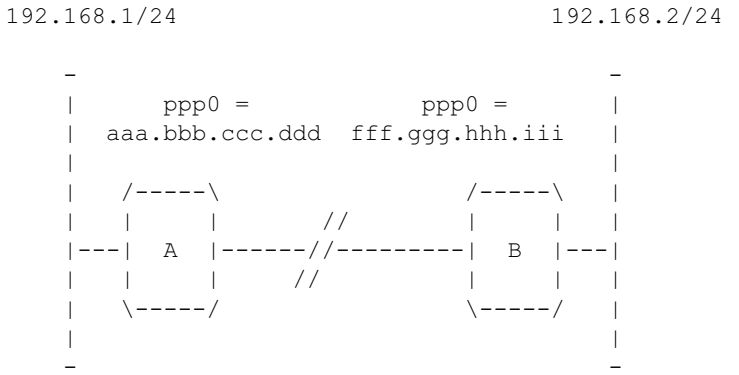
Зачем может понадобится передавать IP-пакеты внутри IP-пакетов? Если Вы никогда не сталкивались с такой потребностью, подобная операция может показаться странной. Два самых главных применения этой техники — Мобильное IP и IP-рассылка. Еще одно применение — Amateur Radio.
Опции компиляции ядра:

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Устройства "IP-тоннели"получают имена 'tunl0', 'tunl1' и т.д.
Так все-таки, зачем это нужно? Обычные правила IP-маршрутизации подразумевают, что IP-сеть имеет адрес и маску. Тем самым, маршрутизация на блок последовательных адресов происходит с помощью одной записи в таблице маршрутизации. Это означает, что при подключении в конкретном месте сети Вы можете иметь конкретный IP-адрес. Если Вы работаете с переносным компьютером, то место Вашего подключения будет постоянно изменяться. Поэтому, если Вы собираетесь временно работать в другом месте, Вы можете настроить машину на Вашем обычном основном месте работы так, чтобы та перенаправляла приходящие на Ваш адрес пакеты на Ваш новый адрес.

6.8.1 Настройка IP-туннеля сеть-сеть.

Рассмотрим сеть следующей структуры:



Эта схема демонстрирует еще один пример использования IPIP-включения — виртуальные частные сети. В этом примере предполагается, что у Вас есть две машины с PPP-подключением к интернету. Каждой из них присвоен IP-адрес. Эти машины подключены к локальным сетям, использующим адреса из зарезервированного диапазона. Предположим, Вы хотите, чтобы машины из одной локальной сети могли взаимодействовать с машинами из другой сети, как будто они соединены непосредственно. Этого можно достичь с помощью IPIP-включения. Это решение, правда, не позволит вашим машинам из внутренних сетей обмениваться данными с другими машинами в интернете — для этого Вам потребуется использовать другие техники вроде IP-маскарада. IPIP-включение производится на машинах A и B — маршрутизаторах.
На машине 'A' выполните команды:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Настройка Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# Настройка ppp0 (запуск ppp, установка маршрута по умолчанию)
pppd
route add default ppp0
#
# Настройка устройства-туннеля
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

А на машине 'B' — команды:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Настройка Ethernet
ifconfig eth0 192.168.2.1 netmask $mask up
route add -net 192.168.2.0 netmask $mask eth0
#
# настройка ppp0 (запуск ppp, установка маршрута по умолчанию)
pppd
route add default ppp0
#
# Настройка устройства-туннеля
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

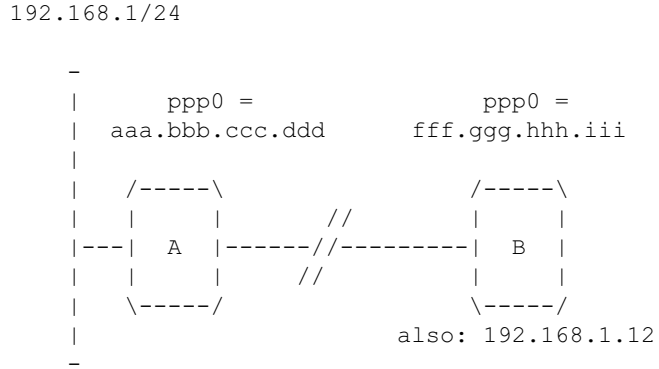
Команда

```
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

расшифровывается как 'Посылать все пакеты для сети `192.168.1.0/24` внутри пакетов, передаваемых на машину по адресу `aaa.bbb.ccc.ddd`'
Обратите внимание, что настройка туннеля требуется с обеих сторон. Устройство-туннель использует параметр '`gw`' команды *route* для определения адреса, на который следует передавать IP-пакеты, с "завернутыми"в них пакетами, предназначенными для сети `192.168.1.0`.

6.8.2 Настройка IP-туннеля сеть-машина.

Совсем не обязательно передавать через туннель данные между двумя сетями. Иногда достаточно, чтобы на одном конце туннеля находилась одна машина. В этом случае настройте устройство '`tunl`' на этой машине на использование "домашнего"адреса, а на маршрутизаторе A используйте маршрут на машину, а не на сеть (еще потребуется использовать механизм кеширования аппаратного адреса (Proxy ARP)). Рассмотрим этот случай. Цель — добиться того, чтобы машина B вела себя как машина, подключенная к интернету, и одновременно как одна из машин сети '`A`.



На маршрутизаторе '`A`' выполните команды:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
```

```
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Настройка Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# настройка ppp0 (запуск ppp, установка маршрута по умолчанию)
pppd
route add default ppp0
#
# Настройка туннеля
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw $remotegw tunl0
#
# Кешировать аппаратный адрес удаленной машины
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

На машине 'B' выполните команды:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Настройка ppp0 (запуск ppp, установка маршрута по умолчанию)
pppd
route add default ppp0
#
# Настройка туннеля
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask $mask gw $remotegwtunl0
```

Такая конфигурация характерна для так называемого "Мобильного IP". Если Вы хотите перемещать одну машину по интернету, сохраняя неизменным IP-адрес. За более подробной информацией о том, как это реализуется на практике, обратитесь к разделу, посвященному мобильному IP.

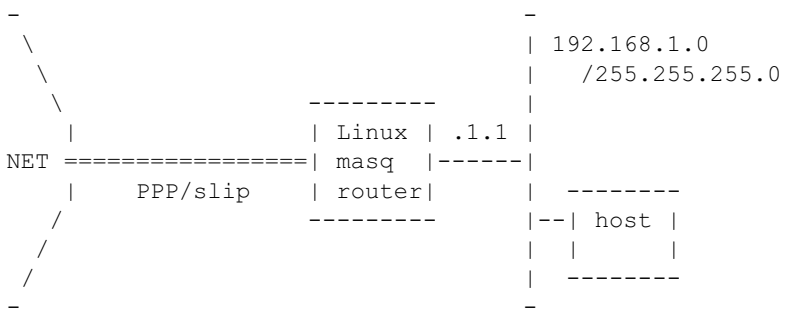
6.9 IP-маскарад (для версий ядра 2.0)

Очень многие имеют обычное сеансовое подключение к интернет, при котором интернет-провайдер выделяет только один IP-адрес. При этом в интернет можно работать только с одной машины. IP-маскарад — трюк, позволяющий нескольким машинам одновременно использовать один IP-адрес, при этом с точки зрения внешних машин выглядеть как одна машина. Правда такая конфигурация работает только "в одну сторону" — маскарадящиеся машины могут обращаться к любым машинам в интернет, но сами при этом остаются недоступными для входящих соединений. Это означает, что некоторые из сетевых сервисов просто не будут работать (например *talk*), а некоторые (например *ftp*)должны быть специально настроены на "пассивный"(PASV) режим работы. К счастью, наиболее распространенные сервисы, такие как *telnet*, *www* и *irc* работают нормально.
Опции компиляции ядра:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
```

```
[*] TCP/IP networking
[*] IP: forwarding/gatewaying
....
[*] IP: masquerading (EXPERIMENTAL)
```

Настройте машину, поддерживающую PPP- или slip-соединение как обычную (не маскарадящую) машину. Кроме того, на этой машине должно быть еще одно сетевое устройство (как правило ethernet), через которое оно подключено к внутренней сети. Настройте эту сеть с использованием адресов из одной из зарезервированных сетей. На всех машинах укажите подключенную к интернет машину в качестве маршрутизатора.
Как правило, сеть имеет такую структуру:



Маршрутизатор настраивается следующими командами:

```
# Маршрутизация для локальной сети
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Маршрут по умолчанию
route add default ppp0
#
# Маскарадить все машины в сети 192.168.1/24
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Для минималистов, или тех, кому лень набивать много команд предлагаем следующую команду, которая будет работать для машины с двумя интерфейсами (все проходящие пакеты должны маскарадится).

```
root# /sbin/ipfwadm -F -a accept -m
```

Более подробную информацию об IP-маскараде вы можете получить на *странице IP-маскарада* <http://www.hwy401.com/achau/ipmasq/>. *Очень* подробным документом об IP-маскараде, в котором описано, как настраивать другие операционные системы на работу с IP-маскарадом Линукса, является "IP-Masquerade mini-HOWTO".

6.10 "Прозрачное"IP-кеширование.

Прозрачное IP-кеширование — возможность Линукса перенаправлять запросы определенных сервисов на других машинах таким же сервисам на локальной машине. Это может быть полезно, когда Ваша машина является маршрутизатором, и одновременно кеширующим сервером. Вы сможете перенаправить все проходящие через маршрутизатор запросы к удаленным машинам на локальный кеш-сервер.
Опции компиляции ядра:

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
        [*] Network firewalls
        ....
        [*] TCP/IP networking
        ....
        [*] IP: firewalling
        ....
        [*] IP: transparent proxy support (EXPERIMENTAL)
```

Настройка прозрачного кеширования производится с помощью программы *ipfwadm*. Пример, который может быть Вам полезен:

```
root# ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

В этом примере все попытки соединения с портом `telnet` (23) на любой удаленной машине будут перенаправлены на локальный порт 2323. На этом порту может работать демон, обрабатывающий telnet-соединения, протоколирующий их и т.п.
Более интересное применение прозрачного кеширования состоит в перенаправлении `http` данных через локальный кеш. К сожалению, протокол, используемый http-кешами отличается от обычного http: Если клиент соединяется с машиной `www.server.com:80` и запрашивает страницу `/path/page`, то при работе с локальным http-кешем он соединяется с машиной `proxy.local.domain:8080` и запрашивает страницу `www.server.com/path/page`.
Для решения этой проблемы существует маленький сервер, называющийся `transproxy`, который Вы можете найти в WWW. Если Вы запустите этот сервер на порту 8081, выполните следующую команду:

```
root# ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Программа `transproxy` будет получать все запросы к удаленным http-серверам и преобразовывать их в запросы к локальному кеш-серверу.

6.11 IPv6

Не успели Вы привыкнуть к правилам работы с протоколом IP, как все изменилось! IPv6 — сокращенное название шестой версии протокола IP. IPv6 был разработан в первую очередь для преодоления проблемы нехватки IP-адресов. Адреса в IPv6 имеют длину 16 байт (128 бит). Кроме того, в IPv6 внесены еще несколько изменений, в основном упрощений, для того чтобы сделать IP-сети более управляемыми.
На данный момент в Линуксе есть работоспособная, хотя еще неполная поддержка IPv6 в ядрах версий `2.1.*`.
Если Вы хотите поэкспериментировать с этой технологией нового поколения, или у Вас есть необходимость использовать ее, прочтите IPv6-FAQ, доступный на *www.terra.net* <http://www.terra.net/ipv6/>.

6.12 Мобильное IP

Под "Мобильным IP"подразумевают способность машины подключаться к интернет из разных мест без изменений в конфигурации. Как правило, при подключении в новом месте Вы получите новый IP-адрес и Вам потребуется переконфигурировать Вашу машину. Мобильное IP решает эту

проблему путем выделения фиксированного IP-адреса и создания туннеля с автоматической маршрутизацией, так чтобы все пакеты, направленные на этот адрес перенаправлялись на реальный IP-адрес, используемый в данный момент.
Существует проект создания полного набора средств мобильного IP для Линукса. Информацию о его текущем состоянии вы можете получить со *страницы мобильного IP в Линуксе* <http://anchor.cs.binghamton.edu/~mobileip/>. Там же находится последняя версия этого пакета.

6.13 IP-рассылка (IP multicast)

IP-рассылка — механизм, позволяющий передавать IP-пакет на несколько машин одновременно. Его используют для "широковещательных"приложений, таких как передача видео- и аудио-информации.
Опции компиляции ядра:

```
Networking options  --->
        [*] TCP/IP networking
        ....
        [*] IP: multicasting
```

Для использования IP-рассылки Вам потребуется набор утилит и небольшая настройка сети. Более подробная информация о IP-рассылке содержится в *Multicast-HOWTO* <Multicast-HOWTO.html>.

6.14 Трансляция сетевых адресов (NAT, Network Address Translation)

Механизм трансляции сетевых адресов — гораздо более стандартизированный "старший брат"IP-маскарада. Он подробно описан в RFC1631. Трансляция адресов предоставляет возможности, которых у IP-маскарада нет и это делает ее более пригодной для использования на маршрутизаторах и файрволлах организаций и в более крупных сетях.
Альфа-версия NAT для ядра версии 2.0.29 написана Михаэлем Хансенстайном (Michael Hasenstein, Michael.Hasenstein@informatik.tu-chemnitz.de). Она (вместе с документацией) доступна со *страницы трансляции IP-адресов* <http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html>
Последние версии ядра 2.1.* включают некоторые из возможностей трансляции адресов в алгоритме маршрутизации.

6.15 Ограничитель потока данных. (Traffic Shaper)

Ограничитель потока данных создает специальные устройства, с ограничениями на передачу данных. Эти устройства являются виртуальными и используют для реальные сетевые устройства для фактической передачи данных. При этом все исходящие IP-пакеты маршрутизируются через устройства-ограничители.
Впервые ограничитель потока появился в ядре версии 2.1.15 и был затем перенесен в ядро версии 2.0.36 (он появился в исправлении 2.0.36-pre-patch-2, распространяемом Аланом Коксом (Alan Cox), автором ограничителя потока и сопровождающим версии ядра 2.0)
На данный момент ограничитель потока может компилироваться в виде модуля и настраивается с помощью программы *shapecfg* примерно следующим образом:

```
shapecfg attach shaper0 eth1
shapecfg speed shaper0 64000
```

Ограничитель контролирует только исходящие IP-пакеты, так как пакеты могут попадать на его интерфейс только в соответствии с таблицами маршрутизации, если Вы хотите ограничивать и

входящий поток данных, Вам потребуется использовать функцию “маршрутизации по адресу отправителя".
В версиях ядра 2.1 такая возможность уже есть, если Вы хотите внести ее и в ядро версии 2.0.*, используйте исправление Майка МакЛагана (Mike McLagan), доступное с `ftp.invlogic.com`. За дальнейшей информацией о работе ограничителя потока данных обратитесь к файлу `Documentation/networking/shaper.txt`, входящему в пакет исходных текстов ядра.
Если Вы хотите испытать тестовую версию ограничителя входящих пакетов, получите пакет `rshaper-1.01` (или более свежую версию) с *ftp.systemy.it* <ftp://ftp.systemy.it/pub/develop>.

6.16 Маршрутизация в ядрах версий 2.2.*

В последних версиях ядра 2.1.* появилось множество нововведений в алгоритме маршрутизации. К сожалению, Вам прийдется дождаться следующей версии этого документа или обратится к исходным текстам ядра.

7 Использование распространенного сетевого оборудования.

7.1 ISDN

"Цифровая сеть интегрированных услуг"(Integrated Services Digital Network (ISDN)) — набор стандартов, определяющих коммутируемую цифровую сеть общего назначения. "Вызов"ISDN создает синхронное соединение точка-точка. ISDN обычно работает на скоростных соединениях, разбитых на некоторое количество "каналов". Существуют 2 типа каналов — каналы типа 'B', через которые передаются пользовательские данные и каналы типа 'D', служащие для передачи управляющей информации. В Австралии, например, ISDN работает, в частности, через каналы с пропускной способностью 2Мбит/с, разбитого на 30 B-каналов с пропускной способностью 64Кбит/с каждый и один 64Кбит/с D-канал. Вы можете использовать любое количество каналов в любых сочетаниях. Например, Вы можете установит 30 соединений 64-килобитных соединений с тридцатью различными точками назначения, либо 15 128-килобитных соединений (по два канала на каждое соединение) или использовать имеющиеся каналы не полностью. Канал можно использовать как для исходящих, так и для входящих соединений. Изначально ISDN предлагалось как средство для телекоммуникационных компаний, которое позволило использовать единый цифровой формат передачи для телефонных соединений, передачи данных без дополнительных настроек.
Существует несколько способов подключить Вашу машину к ISDN. Первый — использовать устройство, называемое "Терминальным Адаптером"(Terminal Adaptor). Это устройство подключается ко входу, установленному Вашим поставщиком ISDN и эмулирует несколько последовательных интерфейсов. Один из этих интерфейсов используется для передачи управляющей информации и установления соединений, а остальные — для передачи данных после того, как соединения будут установлены. В этом случае Линукс будет работать с этими интерфейсами (портами терминального адаптера) как с обычными последовательными устройствами. Второй способ — включение поддержки ISDN в ядро — позволяет установить ISDN-карту в Вашу машину, после чего ядро будет самостоятельно заниматься установлением соединений и работой по ISDN-протоколам.
Опции компиляции ядра:

```
ISDN subsystem  --->
        <*> ISDN support
        [ ] Support synchronous PPP
        [ ] Support audio via ISDN
        < > ICN 2B and 4B support
        < > PCBIT-D support
        < > Teles/NICCY1016PC/Creatix support
```

Реализация ISDN в Линуксе включает в себя поддержку нескольких ISDN-карт. Они перечислены в конфигурации ядра:

- ICN 2B и 4B
- Octal PCBIT-D
- ISDN-карты Teles и совместимые с ними.

Для настройки некоторых из этих карт надо получить у их производителя специальные программы настройки.
Подробная информация о настройке ISDN в Линуксе содержится в каталоге `/usr/src/linux/Documentation/isdn/` и *isdn4linux-FAQ*, доступном с *www.lrz-muenchen.de* <http://www.lrz-muenchen.de/~ui161ab/www/isdn/>. (Включите опцию "english"для получения версии этого документа на английском языке, вместо немецкого.)
Замечание касательно PPP. Протоколы семейства PPP работают как по асинхронным, так и по синхронным каналам. Обычная версия демона *pppd*, входящая в большинство дистрибутивов Линукса, поддерживает только асинхронный режим работы. Для использования PPP поверх ISDN Вам потребуется специальная версия pppd. О том, где взять эту версию, Вы можете прочесть в упомянутой выше документации.

7.2 Протокол PLIP в версиях ядра 2.0.*

Устройствам, работающим по протоколу PLIP присваиваются имена 'plip0', 'plip1 и т.д.
Опции компиляции ядра:

```
Network device support  --->
    <*> PLIP (parallel port) support
```

Протокол *plip* (Parallel line IP) — аналое протокола SLIP, но работающий через параллельные порты Вашей машины (соответствующая разводка кабеля приведена далее в этом документе). Так же как и SLIP, он создает соединение типа *точка-точка*, но обеспечивает более высокую скорость соединения. Кроме того, даже самые обычные принтерные порты можно использовать вместо относительно дорогих карт на основе 16550AFN. Протокол PLIP довольно интенсивно использует процессор по сравнению с последовательным интерфейсом, так что более предпочтительной будет покупка недорогих ethernet-карт, но он вполне пригоден для использования, когда подобной возможности нет. Обычно на хорошо работающем PLIP-соединении можно достичь скорости 20 килобайт в секунду.
PLIP-устройство работает с такой же аппаратурой, что и драйвер параллельного порта, поэтому если Вы хотите их использовать одновременно, Вы должны будете откомпилировать их в виде модулей. После этого Вы сможете указать этим драйверам, какие порты должен использовать каждый из них. За подробностями о том, как компилировать драйвера в виде модулей, обратитесь к "Mudules mini-HOWTO".
К сожалению, на некоторых переносных компьютерах PLIP работать не будет, так как они на них не поддерживаются некоторые используемые plip-интерфейсом комбинации сигналов (эти сигналы не используются принтерами).
Реализация протокола PLIP в Линуксе совместима с драйвером *Crynwyr Packet Driver PLIP*, так что Вы сможете устанавливать plip-соединение между машиной под Линуксом и DOS-машиной и запускать на них любое сетевое программное обеспечение, использующее протокол tcp/ip.
В ядрах версий 2.0.* ядро устанавливает следующее соответствие между plip-устройствами и парами (порт,irq):

```
устройство  порт   IRQ
----------  -----  ---
```

```
plip0       0x3bc  5
plip1       0x378  7
plip2       0x278  2
```

Если Ваш параллельный порт не подходит ни под одну из этих конфигураций, Вы можете использовать команду параметр 'irq' программы *ifconfig*. Убедитесь, что Вы разрешили использование этих IRQ в ROM-BIOS. Другой способ — использовать программу *insmod* с опциями "io=" и "irq=". Например:

```
root# insmod plip.o io=0x288 irq=5
```

Работа PLIP зависит от двух параметров задающих величины ожидания. Значения по умолчанию для этих параметров вполне подойдут в большинстве случаев, необходимость увеличивать их возникает только на очень медленных машинах. При этом увеличивать их следует на **другой** машине, соединенной с медленной. Для изменения этих задержек существует программа *plipconfig*, избавляющая от необходимости перекомпилировать ядро. Она входит во многие дистрибутивы Линукса. Для настройки plip-интерфейса нужно выполнить следующие команды (или добавить их в стартовые скрипты):

```
root# /sbin/ifconfig plip1 localplip pointopoint remoteplip
root# /sbin/route add remoteplip plip1
```

В этом примере используется параллельный порт по адресу 0x378; *localplip* и *remoteplip* — имена или IP-адреса машин, соединенных plip-соединением. Их можно хранить в файле /etc/hosts:

```
# для интерфейса plip
192.168.3.1   localplip
192.168.3.2   remoteplip
```

Параметр *pointopoint* имеет тот же смысл, что и для протокола SLIP — он задает IP-адрес машины на другом конце соединения 'точка-точка'.
Вы можете использовать plip-интерфейс так, как если бы это был интерфейс SLIP, за исключением того, что Вы не сможете (и необходимости в этом не возникнет) использовать программы *dip* и *slattach* применительно к этому интерфейсу.
Дальнейшая информация содержится в "PLIP mini-HOWTO".

7.3 Протокол PLIP в версиях ядра 2.2.*

В процессе разработки версий ядра 2.1.* настройка интерфейса PLIP была несколько изменена.
Опции компиляции ядра:

```
General setup  --->
    [*] Parallel port support
Network device support  --->
    <*> PLIP (parallel port) support
```

Новая реализация plip работает так же, как и старая (используйте те же самые команды *ifconfig* и *route*), однако инициализация устройства будет происходить несколько иначе
"Первое" plip-устройство всегда получает имя "plip0", аналогично тому, как это происходит с ethernet-картами. Реально используемый этим устройством параллельный порт может быть любым из доступных системе параллельных портов (информация о них содержится в каталоге

/proc/parport. Например, если у Вас в машине только один параллельный порт, этот каталог будет содержать единственный подкаталог /proc/parport/0).
Если ядро не смогло самостоятельно определить номер irq, используемый портом, запуск "insmod plip" будет безуспешным. В этом случае запишите нужный номер irq в файл /proc/parport/0/irq и запустите *insmod* еще раз.
Полная информация о работе с параллельными портами содержится в файле Documentation/parport.txt, входящем в исходные тексты ядра.

7.4 PPP

Устройствам PPP ядро присваивает имена 'ppp0', 'ppp1 и т.д. Устройства нумеруются по очереди, первое созданное устройство получает имя ppp0.
Опции компиляции ядра:

```
Networking options  --->
    <*> PPP (point-to-point) support
```

Подробности о настройке PPP приведены в *PPP-HOWTO* <PPP-HOWTO.html>.

7.4.1 Поддержка постоянного ppp-соединения с помощью *pppd*.

Если у Вас (полу-) постоянное подключение к интернет, и Вы хотите, чтобы Ваша машина автоматически восстанавливала его при необходимости — воспользуйтесь программой *pppd*.
Настройте ppp так, чтобы пользователь root мог запускать его командой

```
root# pppd
```

В файле /etc/ppp/options **обязательно** должна присутствовать опция '-detach'. После этого добавьте строчку

```
pd:23:respawn:/usr/sbin/pppd
```

в файл /etc/inittab (после строк запуска демонов *getty*). Теперь процесс init будет перезапускать *pppd*, если тот "умрет".

7.5 SLIP-клиент

Устройствам SLIP ядро присваивает имена 'sl0', 'sl1 и т.д. Устройства нумеруются по очереди, первое созданное устройство получает имя sl0.
Опции компиляции ядра:

```
Network device support  --->
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

SLIP (Serial Line Internet Protocol, Протокол интернет-соединений по последовательным линиям) позволяет использовать для передачи tcp/ip последовательные линии, будь то телефонная линия с подключенным модемом или какая-нибудь выделенная линия. Для использования протокола SLIP Вам необходим доступ к SLIP-серверу. Такие сервера доступны во многих учебных и коммерческих учреждениях. p> Протокол SLIP использует для передачи IP-пакетов последовательный порт. Для этого он должен иметь доступ к соответствующему последовательному устройству. Для того, чтобы установить соответствие между slip-интерфейсами и последовательными портами используется механизм *ioctl-вызовов* (i/o control). Настройка выполняется с помощью программ *dip* и *slattach*.

7.5.1 dip

dip (Dialup IP) — весьма интеллектуальная программа, позволяющая устанавливать скорость работы последовательного порта, управлять модемом, автоматически регистрироваться на удаленной машине после соединения, принимать сообщения от SLIP-сервера и извлекать из них информацию о выделенном IP-адресе и выполнять необходимые ioctl-вызовы для переключения последовательного порта в SLIP-режим. Программа *dip* позволяет самостоятельно писать скрипты для автоматизации подключения к удаленной машине.
Вы можете получить эту программу с *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/dip/dip337o-uri.tgz>.
Для установки выполните следующие команды:

```
user% tar xvzf dip337o-uri.tgz
user% cd dip-3.3.7o
user% vi Makefile
root# make install
```

В файле `Makefile` предполагается, что у Вас на машине заведена группа *uucp*, но Вы можете при необходимости изменить ее на *dip* или *SLIP*.

7.5.2 slattach

В противоположность *dip*, программа *slattach* достаточно проста и легка в использовании, но не имеет всех возможностей *dip*. Эта программа не поддерживает скриптов, все что она делает — это настраивает последовательный порт. Предполагается, что Вы знаете все необходимые параметры Вашего соединения и Вы уже установили последовательное соединение с сервером перед тем как запустить *slattach*. Эта программа идеально подходит в случае постоянного подключения к серверу по выделенной линии.

7.5.3 В каких случаях использовать какую программу?

Советуем Вам использовать программу *dip* если Вы соединяетесь с сервером с помощью модема по телефонной линии или у Вас другой тип непостоянного соединения. В случае постоянного соединения по выделенной линии (не требуется специальных действий для того, чтобы установить последовательное соединение) используйте программу *slattach*. Второй вариант подробно описан в разделе 'Постоянное SLIP-соединение'.
Настройка slip-интерфейса во многом похожа на настройку ethernet (СМ. раздел 'Ethernet'), однако есть несколько важных отличий.
Во-первых, в slip-соединении участвуют только 2 машины. В отличие от ethernet-сети, которая доступна все время, slip-соединение может потребоваться специальным образом восстанавливать в ходе работы.
Если Вы используете *dip*, slip-соединение устанавливается не в момент загрузки, а несколько позже, когда Вам потребуется реально использовать это соединение (эту процедуру можно автоматизировать). Если Вы используете *slattach*, Вам потребуется добавить блок установки соединения в файл *rc.inet1*. Подробности этого следуют ниже.
SLIP-сервера делятся на 2 типа — сервера с динамическим выделением адресов (динамические сервера) и сервера со статическим выделением адресов (статические сервера). Почти любой SLIP-сервер предложит Вам ввести имя регистрации и пароль для установки соединения. Программа *dip* производит регистрацию автоматически.

7.5.4 Сеансовое соединение, Статический SLIP-сервер и DIP.

Статический SLIP-сервер выделяет Вам один и тот же IP-адрес при любом соединении. Каждый раз, подключаясь к серверу, Вы настраиваете slip-интерфейс на использование этого адреса. Статиче-

ский SLIP-сервер ответит на Ваш модемный вызов, предложит ввести имя регистрации и пароль, а затем будет маршрутизировать IP-пакеты, адресованные на Ваш адрес через созданный slip-интерфейс. В этом случае Вы можете добавить соответствующие строчки в файлы `/etc/hosts`, `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` и `rc.local`. Обратите внимание, что редактировать файл `rc.inet1` не нужно — настройка интерфейса будет целиком произведена программой *dip*. Все что Вам нужно указать ей всю необходимую информацию о Вашем последовательном порте.
Можете перейти к разделу 'Использование программы Dip' для того, чтобы настроить *dip*.

7.5.5 Сеансовое соединение, Динамический SLIP-сервер и DIP.

Динамический SLIP-сервер выделяет Вам при каждом соединении случайный IP-адрес из набора доступных. Это означает, что нет никакой гарантии, что данный момент времени Ваша машина имеет некоторый фиксированный IP-адрес, и что адрес, который Вы использовали будет использован кем-то другим после того как Вы прервали соединение. Администратор SLIP-сервера выделяет набор адресов, и при каждой регистрации сервер находит первый неиспользованный адрес, регистрирует пользователя, выделяет ему этот адрес и посылает сообщение, содержащее этот адрес.
Настройка для работы с таким типом сервера отличается только тем, что Вам нужно получить от сервера IP-адрес и настроить slip-интерфейс в соответствии с этим.
Как и в предыдущем случае, программа *dip* выполнит все необходимые действия, последние версии самостоятельно извлекают IP-адрес и настраивают slip-интерфейс.
Можете перейти к разделу 'Использование программы Dip' для того, чтобы настроить *dip*.

7.5.6 Использование программы DIP.

Как уже было сказано ранее *dip* — мощная программа, которая может упростить и автоматизировать процесс соединения со SLIP-сервером, регистрации на нем и настройки slip-интерфейса с помощью программ *ifconfig* и *route*.
Для использования *dip* пишется 'dip-скрипт' — набор команд, понимаемых *dip*. Эти команды указывают как выполнять все необходимые для установки соединения процедуры. Для того, чтобы понять основную идею, можете взглянуть на файл `sample.dip`, идущий в комплекте с программой *dip*. *dip* имеет множество опций, с которыми можно ознакомится на man-странице, файле README и файлах-примерах из пакета *dip*.
p>Вы могли заметить, что в скрипте `sample.dip` предполагается, что Вы соединяетесь со статическим SLIP-сервером, и выделенный Вам IP-адрес известен заранее. Для динамических SLIP-серверов в последних версиях программы *dip* предусмотрена возможность автоматически считывать и использовать выделенный динамически IP-адрес. Приводящийся ниже пример — модифицированный файл `sample.dip` из пакета *dip337j-uri.tgz*. Он может послужить хорошей отправной точкой. Можете скопировать его в `/etc/dipscript` и отредактировать в соответствии с Вашей конфигурацией.

```
#
# sample.dip    Dialup IP connection support program.
#
#               Этот файл иллюстрирует использование программы DIP.
#       Он был опробован на динамических серверах типа Annex, если Вы
#       используете статический сервер, используйте файл sample.dip из
#       пакета dip337-uri.tgz .
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#
```

```
main:
# Установка имени и адреса машины-сервера.
# В данном примере это 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Установка маски на интерфейсе sl0 в 255.255.255.0
netmask 255.255.255.0
# Выбор последовательного порта и скорости соединения.
port cua02
speed 38400

# Сброс модема.
# Иногда приводит к проблемам!
reset

# "Стандартные" предопределенные значения переменной "errlevel":
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# Можете изменить их в функции "addchat()"

# Подготовка к дозвонке.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# Соединение установлено.  Регистрация.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# Регистрация завершена.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Перевести сервер в режим SLIP
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Получение IP-адрес от сервера.
#   Здесь предполагается, что после перехода в режим SLIP сервер сообщает
#   выделенный IP-адрес
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Установка параметров протокола SLIP.
get $mtu 296
# Выполнить "route add -net default xs4all.hacktic.nl"
```

```
default

# Завершение
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

В приведенном примере предполагается, что вы соединяетесь с динамическим SLIP-сервером. Если Вы используете статический сервер, используйте файл `sample.dip` из пакета *dip337-uri.tgz*.
Когда *dip* обрабатывает команду *get $local*, он ищет в полученных данных строку, которая выглядит как IP-адрес, то есть числа, разделенные символами '.'. Эта возможность была внесена специально для работы с динамическими SLIP-серверами. Она дает возможность автоматизировать процесс настройки на неизвестный заранее адрес.
В приведенном примере маршрутизация по умолчанию производится через slip-соединение, если для Вашей конфигурации это не подходит, уберите команду default из скрипта. Если Вы запустите программу *ifconfig* после окончания работы скрипта, Вы обнаружите устройство *sl0*. Это Ваш slip-интерфейс. При необходимости Вы можете его перенастроить с помощью программ *ifconfig* и *route* после того как программа *dip* завершится.
Программа *dip* позволяет выбрать много разных протоколов работы с помощью команды `mode`, наиболее распространенный пример — протокол сжатия данных *cSLIP*. Обе машины, участвующие в slip-соединении должны установить одинаковый протокол работы.
Приведенный пример достаточно надежен и корректно обработает большинство возможных ошибок при соединении. Более подробная информация содержится на man-странице программы *dip*. Кроме того, Вы можете написать скрипт который будет пытаться перезванивать несколько раз в течении заданного периода времени или даже перебирать по очереди доступные SLIP-сервера, если их несколько.

7.5.7 Постоянное SLIP-соединение по выделенной линии и программа slattach.

Если у вас есть выделенная линия или прямой кабель между двумя машинами, или какой-либо другой тип постоянного последовательного соединения, у Вас нет необходимости вникать во все сложности использования программы *dip*. Очень простая в использовании программа *slattach* позволит Вам настроить slip-соединение.

Поскольку речь идет о постоянном соединении, Вам потребуется добавить несколько команд в файл `rc.inet1`. Фактически, все что Вам нужно сделать — это настроить скорость работы последовательного интерфейса и перевести его в режим SLIP. Именно для этого и предназначена программа *slattach*. Добавьте следующие строки в файл `rc.inet1`:

```
#
# Установка статического slip-соединения по выделенной линии
#
# настройка /dev/cua0 на скорость 19.2kbps и протокол cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
```

Где:

IPA.IPA.IPA.IPA

Ваш IP-адрес.

IPR.IPR.IPR.IPR

IP-адрес машины на другом конце соединения.

Программа *slattach* выделяет первый неиспользуемый slip-интерфейс для работы через указанное последовательное устройство, начиная с *sl0*. Таким образом, первый вызов *slattach* связывает указанное устройство с интерфейсом *sl0*, второй вызов — с интерфейсом *sl0*, и т.д.
slattach позволяет указывать разные протоколы соединения с помощью опции `-p`. Как правило это протоколы *SLIP* или *cSLIP*, в зависимости от того, используете ли Вы сжатие данных или нет. Внимание! Обе связывающиеся машины должны использовать один и тот же протокол.

7.6 SLIP-сервер.

Если у Вас есть подключенная к сети машина, через которую Вы хотите подключатся с других машин по последовательным линиям, Вы должны настроить эту машину как сервер. Если Вы выбрали в качестве протокола SLIP у Вас есть три варианта такой настройки. Автор предпочитает пользоваться первым — программой *sliplogin*, самой легкой в настойке и освоении, но будут описаны и другие, так что Вы можете самостоятельно сделать выбор.

7.6.1 SLIP-сервер на базе программы *sliplogin*.

sliplogin — программа, которую Вы можете использовать вместо командного интерпретатора для пользователей, которые подключаются к Вашему серверу по протоколу SLIP. Она позволяет Вашей машине быть как статическим (пользователю всегда выделяется фиксированный IP-адрес), так и динамическим (пользователь получает при регистрации первый свободный IP-адрес из заданного набора) SLIP-сервером. *sliplogin* преобразует терминал в slip-соединение.
Пользователь регистрируется так же как и при локальной регистрации, вводя имя регистрации и пароль, но вместо обычного командного интерпретатора запускается *sliplogin*. Она ищет в файле `/etc/slip.hosts` строку, соответствующую введенному имени регистрации. Если такая строка находится, *sliplogin* настраивает последовательную линию на режим 8 бит без стоп-битов и переводит ее в режим SLIP. После этого запускается скрипт, который настраивает slip-интерфейс на использование нужного IP-адреса, маски и настраивает маршрутизацию. Этот скрипт обычно называется `/etc/slip.login`, но как и в случае с *getty*, Вы можете для некоторых пользователей создать специальные инициализационные скрипты с именами `/etc/slip.login.loginname`, которые запустятся при регистрации этого пользователя.
Для полной настройки программы *sliplogin* Вам потребуется отредактировать 4 (иногда 5) файла. Это файлы:

- `/etc/passwd`, для создания пользователей.
- `/etc/slip.hosts`, содержащий информацию, специфичную для каждого из slip-пользователей.
- `/etc/slip.login`, настраивающий маршрутизацию для пользователя после регистрации.
- `/etc/slip.tty`, этот файл нужен только в случае динамического SLIP-сервера. Он содержит список допустимых IP-адресов
- `/etc/slip.logout` — содержащий команды, которые надо выполнить после отключения пользователя.

Где взять *sliplogin* Пакет *sliplogin* входит во многие дистрибутивы, поэтому он уже может быть установлен на Вашей машине. Если это не так, Вы можете получить его с *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/linux/system/Network/serial/sliplogin-2.1.1.tar.gz>. В состав пакета входят исходные тексты, откомпилированные программы и man-страница.
Для того, чтобы доступ к *sliplogin* имели только нужные пользователи, добавьте в файл `/etc/group` подобную строку:

```
 ..
slip::13:radio,fred
 ..
```

Когда Вы устанавливаете пакет *sliplogin*, группой-владельцем программы *sliplogin* устанавливается группа `slip`, и только пользователи из этой группы могут запускать *sliplogin*. В приведенном примере запускать *sliplogin* смогут только пользователи `radio` и `fred`.
Для установки пакета используйте следующие команды:

```
user% cd /usr/src
user% gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
user% cd sliplogin-2.1.1
user% <..отредактируйте файл Makefile если Вы не используете механизм теневых паролей ..>
root# make install
```

Если Вы хотите перекомпилировать программу перед установкой, добавьте команду `make clean` перед `make install`. Если Вы хотите установить *sliplogin* не в каталог /sbin, а в какой-либо другой, отредактируйте правило *install* в файле `Makefile`.
Дополнительная информация содержится в файле `README`, входящем в состав пакета.

Настройка файла `/etc/passwd` для работы со SLIP. Как правило, Вам потребуется завести в файле `/etc/passwd` специальных пользователей, для тех, кто использует Вашу машину в качестве SLIP-сервера. Повсеместно используется следующее правило присвоения имен: имя пользователя получается из заглавной 'S' и имени машины. Например, если с Вашим сервером соединяется машина `radio`, то в файле `/etc/passwd` будет запись следующего вида:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

В целом не очень важно, как будет называться пользователь, главное чтобы имена не создавали путаницы.
Замечание: Для этих пользователей не нужно создавать "домашние каталоги", в качестве такого каталога можно указать `/tmp`. Обратите внимание, что в качестве командного интерпретатора используется *sliplogin*.

Файл /etc/slip.hosts *sliplogin* ищет в этом файле информацию, которая относится к зарегистрировавшемуся пользователю. В этом файле указывается IP-адрес и маска, которые будут присвоены пользователю после регистрации. Например, информация для машины `radio`, которая получает статический IP-адрес и машины `albert`, которая получает динамический IP-адрес может выглядеть так:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Строки в файле `/etc/slip.hosts` делятся на следующие поля:

1. имя регистрации пользователя.
2. IP-адрес сервера, т.е. этой машины.
3. IP-адрес, который надо присвоить пользователю. Если это поле имеет значение `DYNAMIC` то IP-адрес будет выделен на основе информации из файла `/etc/slip.tty` (СМ. ниже). **Внимание!** Вы должны использовать версию sliplogin не менее 1.3 для того чтобы пользоваться этой возможностью.
4. маска, присваиваемая пользователю в десятичной записи. Например 255.255.255.0 для маски сети класса C.
5. Режим протокола SLIP при работе с этим пользователем. Допустимые значения "`normal`", "`compressed`"и некоторые другие.
6. Временная задержка, в течении которой программа будет ожидать данных. Если в течении этого периода не будет передано или получено ни одного IP-пакета соединение будет автоматически разорвано. Если Вы не хотите, чтобы это происходило, задайте отрицательное значение этого параметра.
7. дополнительные параметры.

Замечание: В полях 2 и 3 Вы можете использовать IP-адреса, так и имена. Если Вы используйте имена, убедитесь что эти имена удается преобразовать в IP-адреса, иначе регистрация не выполнится. Для проверки попробуйте запустить telnet на машину с нужным именем, если Вы увидите сообщение вида '*Trying nnn.nnn.nnn...*' то преобразование было успешным — Ваша машина смогла определить IP-адрес для этой машины. Иначе Вы получите сообщение '*Unknown host*' — в этом случае задайте IP-адрес явно или проверьте настройки системы преобразования имен (СМ. раздел `Система преобразования имен`).
Чаще всего используются режимы SLIP

normal

SLIP без сжатия данных.

compressed

сжатие заголовков пакетов (cSLIP)

Эти два режима взаимоисключающие — Вы не можете использовать их одновременно. Остальные опции описаны на man-странице.

Файл /etc/slip.login. Если *sliplogin* нашел в файле /etc/slip.hosts подходящую запись, он пытается выполнить скрипт /etc/slip.login. Этот скрипт настраивает slip-интерфейс на сервере. Пример файла /etc/slip.login, включенный в пакет *sliplogin* выглядит так:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# инициализационный скрипт для SLIP-сервера. Параметры:
#     $1              $2       $3    $4, $5, $6 ...
#   имя_интерфейса  скорость    pid   параметры из файла slip.host
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Этот скрипт использует программы *ifconfig* и *route* для настройки slip-интерфейса, точно так же, как это делает программа *slattach*.
Кроме того последняя команда скрипта создает ARP-запись, для того, чтобы машины в локальной сети сервера могли соединятся с удаленной машиной. В поле <hw_addr> вы должны указать аппаратный адрес ethernet-карты на сервере. Если сервер не подключен к локальной сети, эту команду можно опустить.

Файл /etc/slip.logout. Когда последовательное соединение завершается, нужно вернуть slip-устройсво в исходное состояние, так чтобы следующие slip-пользователи могли нормально регистрироваться. Для этого используется файл /etc/slip.logout. Это скрипт, вызывающийся с теми же параметрами, что и /etc/slip.login.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

В данном примере он дезактивирует интерфейс и удаляет arp-запись. Если сервер не подключен к локальной сети, запуск *arp* можно опустить.

Файл /etc/slip.tty. Если Ваш сервер поддерживает динамическое выделение адресов и некоторые из записей в файле /etc/slip.hosts имеют значение DYNAMIC в поле IP-адреса, то Вы должны создать файл /etc/slip.tty, содержащий список адресов для каждого из портов.
Файл /etc/slip.tty содержит список устройств *tty*, настроенных на работу по протоколу SLIP и IP-адреса, которые надо присваивать пользователю при соединении через эти устройства.
Пример такого файла:

```
# slip.tty    список соответствий tty -> IP-адрес
# формат: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Этот файл указывает что при соединении через устройство `/dev/ttyS0` для пользователей с полем `DYNAMIC` в файле `/etc/slip.hosts` будет присвоен адрес `192.168.0.100`.
При такой системе Вам достаточно выделить по одному адресу для каждого последовательного порта, при этом количество потраченных адресов будет минимальным.

7.6.2 SLIP-сервер на базе программы *dip*.

Часть приведенной в этом разделе информации взята с man-страницы программы *dip*, на которой вкратце описано ее использование в качестве slip-сервера. Кроме того эта информация относится к версии *dip337o-uri.tgz* пакета *dip*, в других версиях возможны изменения.
В *dip* есть "входной" режим работы, при котором автоматически отыскивается запись в файле `/etc/diphosts` о пользователе, который запустил *dip*, и в соответствии с ней настраивается интерфейс и соединение переводится в режим SLIP. Чтобы запустить *dip* в таком режиме, нужно выполнить команду *diplogin*. Чтобы использовать *dip* в качестве SLIP-сервера, нужно создать пользователей, у которых в качестве командного интерпретатора указан *diplogin*.
Сначала Вы должны создать ссылку:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

После этого добавьте записи в файлы `/etc/passwd` и `/etc/diphosts`. Чтобы использовать *dip* в качестве SLIP-сервера создайте пользователей с командным интерпретатором *diplogin*. Обычно имена этих пользователей делают начинающимися на заглавное 'S', например 'Sfredm'. Строка в файле `/etc/passwd` для такого пользователя будет выглядеть так:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^          ^^         ^^  ^^    ^^    ^^    ^^
|           |          |   |     |     |     \__ командный интерпретатор diplogin
|           |          |   |     |     \________ Домашний каталог
|           |          |   |     \______________ Полное имя
|           |          |   \____________________ ID группы
|           |          \________________________ ID пользователя
|           \___________________________________ Зашифрованный пароль
\_______________________________________________ имя пользователя
```

После того, как программа *login* успешно зарегистрирует такого пользователя будет выполнена программа *diplogin*. *dip* при таком запуске переходит во "входной" режим, и вызывает функцию *getuid()*, чтобы определить текущего пользователя. После этого в файле `/etc/diphosts` ищется первая подходящая строка с нужным именем пользователя или *tty* устройством и *dip* выполняет все нужные настройки. Помещая запись о пользователе в файл `diphosts` Вы создаете для него специальную конфигурацию для остальных пользователей будет выбрана конфигурация по умолчанию. Таким образом вы можете создавать пользователей со статическим или динамическим адресом.
dip автоматически создает arp-запись при работе во "входном" режиме, так что Вам не нужно беспокоится создании и удалении этих записей вручную.

Файл `/etc/diphosts`. Файл `/etc/diphosts` используется программой *dip*. В этом файле хранятся настройки для удаленных машин. Это могут быть машины, которые используют Вашу машину в качестве сервера или машины, которые Вы используете в качестве сервера. Формат этого файла следующий:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Каждая строка состоит из нескольких полей:

1. имя пользователя: имя, возвращаемое getpwuid(getuid()) или имя tty.
2. не используется: для совместимости с passwd
3. Удаленный адрес: IP-адрес удаленной машины или ее имя
4. Local Address: IP-адрес этой машины или ее имя
5. Маска: IP-маска в десятичной записи
6. Комментарий: можете вносить сюда свои примечания.
7. протокол: Slip, CSlip и др.
8. MTU: десятичное число

Пример такой строки:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

Эта строка определяет соединение с машиной по адресу 145.71.34.1 по протоколу SLIP с mtu 296. Или:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

Эта строка определяет соединение с машиной по адресу 145.71.34.2 по протоколу cSLIP с mtu 1006. Все пользователи, которым адрес выделяется статически, должны иметь свои строчки в файле /etc/diphosts. Если Вы хотите, чтобы адрес присваивался динамически — создайте запись для tty-устройства и не создавайте записи для этого пользователя. Обязательно создайте по одной записи для каждого tty-устройства, которое используется для доступа по протоколу SLIP, чтобы пользователь всегда получил настроенное соединение независимо от того, по какой линии он дозвонился.
Когда пользователь устанавливает соединение, ему предлагают ввести имя и пароль как при обычной локальной регистрации. Пользователь должен ввести slip-имя и пароль. Если регистрация проходит успешно, пользователю будет достаточно переключится в slip-режим. При этом соединение со стороны сервера будет настроено в соответствии с информацией из файла diphosts.

7.6.3 SLIP-сервер на базе пакета *dSLIP*.

Матт Дилон (Matt Dillon <dillon@apollo.west.oic.com>) написал пакет, который позволяет Вашей машине быть slip-сервером и slip-клиентом. Этот пакет состоит из множества маленьких программ и скриптов, управляющих slip-соединением. Для нормальной работы пакета Вы должны установить пакет *tcsh*. В состав *dSLIP* включена программа *expect*, и Вам возможно потребуется небольшой опыт работы с этой программой для того, чтобы настроить пакет под свои нужды. Это несложно, так что не пугайтесь.
Файл README содержит очень хорошие инструкции по установке пакета, так что нет смысла повторять их здесь.
Пакет *dSLIP* можно получить с домашнего сайта *apollo.west.oic.com* <ftp://apollo.west.oic.com/pub/linux/dillon_src/dSLIP203.tgz> или с *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/dSLIP203.tgz>.
Прочтите файл README, создайте нужные записи в файлах /etc/passwd и etc/group и выполните команду make install.

8 Другие сетевые технологии.

Этот раздел посвящен специальным (не очень распространенным) сетевым технологиям. Подразделы в нем независимы друг от друга. Технологии описаны в алфавитном порядке.

8.1 ARCNet

Устройствам типа ARCNet присваиваются имена 'arc0e', 'arc1e', 'arc2e' и т.д. или 'arc0s', 'arc1s', 'arc2s' и т.д. Первая обнаруженная ядром сетевая карта получает имя 'arc0e' или 'arc0s', все оставшиеся нумеруются по порядку обнаружения. Последняя буква в имени устройства означает, выбран ли режим ethernet-пакетов или режим, описанный в RFC1051.
Опции компиляции ядра:

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

После того как Ваше ядро будет откомпилировано с поддержкой Вашей карты, Вам достаточно выполнить простые команды настройки следующего вида:

```
root# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

За более подробной информацией обратитесь к файлам /usr/src/linux/Documentation/networking/arc
и /usr/src/linux/Documentation/networking/arcnet-hardware.txt
Поддержка ARCNet была разработана Эвери Пеннераном (Avery Pennarun, apenwarr@foxnet.net).

8.2 Appletalk (AF_APPLETALK)

Протокол AppleTalk использует уже существующее сетевое устройство и не создает новых имен.
Опции компиляции ядра:

```
Networking options  --->
    <*> Appletalk DDP
```

Поддержка протокола AppleTalk позволяет Вашей машине работать в сетях фирмы Apple. Вы можете совместно использовать диски и принтеры на машинах Apple и машинах под Линуксом. Для этого вам потребуется программный пакет *netatalk*. Уэсли Крэйг (Wesley Craig netatalk@umich.edu) работает в группе 'Research Systems Unix Group' в Университете штата Мичиган, которая создала этот пакет. Возможно, пакет *netatalk* уже есть в Вашем дистрибутива Линукса, либо Вы можете получить его с *ftp-сайта Мичиганского Университета* <ftp://terminator.rs.itd.umich.edu/unix/netatalk/>
Для компиляции и установки пакета выполните следующие команды:

```
user% tar xvfz .../netatalk-1.4b2.tar.Z
user% make
root# make install
```

При желании Вы можете откорректировать файл 'Makefile' перед запуском *make* — например изменить значение переменной DESTDIR, которая определяет, в какой каталог будут установлены файлы пакета. Обычно вполне подходит значение по умолчанию /usr/local/atalk.

8.2.1 Настройка AppleTalk.

Первым делом убедитесь, что в файле /etc/services есть сроки вида

```
rtmp  1/ddp   # Routing Table Maintenance Protocol
nbp   2/ddp   # Name Binding Protocol
echo  4/ddp   # AppleTalk Echo Protocol
zip   6/ddp   # Zone Information Protocol
```

Затем создайте конфигурационные файлы в каталоге /usr/local/atalk/etc (либо в подкаталоге etc каталога, в который Вы установили пакет).
Сперва создайте файл /usr/local/atalk/etc/atalkd.conf. Изначально этот файл должен содержать только имя сетевого устройства, через которое работает протокол AppleTalk:

```
eth0
```

Демон AppleTalk после запуска добавит другие данные в этот файл.

8.2.2 Предоставление файловой системы для использования.

Вы можете предоставить другим машинам в AppleTalk-сети возможность использовать файлы на Вашей машине (экспортировать свою файловую систему). Для этого отредактируйте файл /usr/local/atalk/etc/AppleVolumes.system . Есть еще один файл, /usr/local/atalk/etc/AppleVolumes.default, имеющий такой же формат, в котором описано, какие права будут предоставлены "непривилегированым"пользователям (guest).
Полностью формат этих файлов описан на man-странице программы afpd. Простейший пример может выглядеть так:

```
/tmp Scratch
/home/ftp/pub "Public Area"
```

В такой конфигурации ваш каталог /tmp будет виден под именем 'Scratch', а каталог /home/ftp/pub — под именем 'Public Area'. Имена можно опускать, в этом случае демон присвоит им значения по умолчанию.

8.2.3 Предоставления принтера для использования.

Для того, чтобы дать возможность другим машинам в сети использовать Ваш принтер Вам достаточно запустить демона papd. Он будет принимать сетевые запросы на печать и передавать данные локальному демону печати. Для настройки демона *papd* отредактируйте файл /usr/local/atalk/etc/papd.conf. Формат этого файла совпадает с форматом файла /etc/printcap. Имя, которое Вы укажете в этом файле будет именем принтера в сети. Например:

```
TricWriter:\
   :pr=lp:op=cg:
```

Такой файл конфигурации создаст в сети принтер с именем 'TricWriter'. Все запросы к этому принтеру будут перенаправляться на локальный принтер lp (согласно информации из файла /etc/printcap) и печататься м помощью демона *lpd*. Опция 'op=cg' задает имя пользователя (cg), который является оператором данного принтера.

8.2.4 Запуск AppleTalk.

Пришло время проверить сделанные настройки. В состав пакета *netatalk* входит файл `rc.atalk`, как правило, достаточно запустить его.

```
root# /usr/local/atalk/etc/rc.atalk
```

Этот файл запустит всех необходимых демонов и будет по мере запуска выдавать на консоль сообщения о своей работе.

8.2.5 Проверка AppltTalk.

Для того, чтобы проверить работоспособность запущенных программ, на одной из машин Apple выберите пункт Chooser в главном меню, и щелкните мышью по пункту AppleShare. Вы должны увидеть Вашу машину.

8.2.6 Особенности работы AppleTalk.

- Вам может потребоваться запустить AppleTalk перед запуском IP-служб. Если у Вас возникают проблемы при запуске программ AppleTalk, или проблемы с IP-сетью, попробуйте запустить AppleTalk перед запуском `/etc/rc.d/rc.inet1` (файла, стартующего IP-сеть).
- Демон *afpd* создает на диске множество служебных файлов. В каждом из экспортированных каталогов он создает каталоги `.AppleDesktop` и `Network Trash Folder`. Кроме того, в каждом из подкаталогов экспортированных каталогов создается каталог `.AppleDouble` для хранения ресурсов файлов. Так что тщательно подумайте, прежде чем экспортировать корневой каталог `/`.
- Демон *afpd* получает с машин Apple пароли в нешифрованном виде, что может привести к проблемам с безопасностью. Запуская этого демона на машине, доступной из интернета, Вы подвергаете себя риску.
- Диагностические программы, такие как *ifconfig* и *netstat* не поддерживают протокол AppleTalk. Необработанную информацию о работе этого протокола Вы можете посмотреть в каталоге `/proc/net/`.

8.2.7 Дополнительная информация.

За более подробной информацией о настройке и работе с AppleTalk Вы можете обратится на web-страницу *Netatalk-HOWTO* Андерса Браунворса (Anders Brownworth) на сервере *thehamptons.com* <http://thehamptons.com/anders/netatalk/>.

8.3 ATM

Проект по поддержке протокола ATM (Asynchronous Transfer Mode, Асинхронный Режим Передачи) ведется Вернером Альмесбергером (Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>) Информацию о текущем состоянии проекта можно получить на *lrcwww.epfl.ch* <http://lrcwww.epfl.ch/linux-atm/>.

8.4 AX25 (AF_AX25)

Устройства типа AX.25 имеют имена 'sl0', 'sl1' и т.д. в ядрах версии 2.0.* и имена 'ax0', 'ax1' и т.д. в ядрах версии 2.1.*.
Опции компиляции ядра:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

Протоколы AX25, Netrom и Rose рассмотрены подробно в *AX25-HOWTO* <AX25-HOWTO.html>. Эти протоколы используются операторами 'Amateur Radio'.
Большая часть работы по реализации этих протоколов в Линуксе была выполнена Джонатаном Нейлором (Jonathon Naylor, jsn@cs.nott.ac.uk).

8.5 DECNet

Поддержка DECNet находится в процессе разработки и должна появится в новых версиях ядра 2.1.*.

8.6 FDDI

Устройства FDDI получают имена 'fddi0', 'fddi1', 'fddi2' и т.д. Первая обнаруженная ядром карта FDDI получает имя 'fddi0', остальные нумеруются в порядке обнаружения.
Лоуренс В. Стефани (Lawrence V. Stefani, larry_stefani@us.newbridge.com) написал драйвер для FDDI-карт для шин EISA и PCI производства фирмы Digital Equipment Corporation.
Опции компиляции ядра:

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

После того как вы откомпилируете ядро с поддержкой FDDI, Вы должны настроить fddi-интерфейс. Настройка выполняется аналогично ethernet-картам. Вам понадобится заменить имя ethernet-интерфейса на имя fddi-интерфейса в командах *ifconfig* и *route*.

8.7 Сети с ретрансляцией кадров (Frame Relay).

Устройства, работающие по протоколу ретрансляции кадров получают имена 'dlci00', 'dlci01' и т.д. для устройств типа DLCI или 'sdla0', 'sdla1' и т.д. для устройств типа FRAD.
Ретрансляция кадров — новая технология, призванная обеспечивать передачу данных с переменной интенсивностью потока. Вы подключаетесь к сети с ретрансляцией кадров с помощью устройства FRAD (Frame Relay Access Device, Устройство Доступа к сети с Ретрансляцией Кадров). Линукс поддерживает передачу IP-пакетов через сеть с ретрансляцией кадров в соответствии с RFC1490.
Опции компиляции ядра:

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Майк МакЛаган (Mike McLagan, mike.mclagan@linux.org), разработал драйвера и утилиты поддержки сетей с ретрансляцией кадров.
На текущий момент единственными поддерживаемыми устройствами FRAD являются S502A, S502E и S508 фирмы *Sangoma Technologies* <http://www.sangoma.com/>.
Для настройки устройств FRAD и DLCI Вам потребуются утилиты настройки, которые Вы можете получить по ftp с *ftp.invlogic.com* <ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz>.
Компиляция и установка этого пакета несколько осложнены из-за отсутствия "главного"Makefile :

```
user% tar xvfz .../frad-0.15.tgz
user% cd frad-0.15
user% for i in common dlci frad; make -C $i clean; make -C $i; done
root# mkdir /etc/frad
root# install -m 644 -o root -g root bin/*.sfm /etc/frad
root# install -m 700 -o root -g root frad/fradcfg /sbin
root# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Приведенные команды рассчитаны на интерпретатор *sh*. Если Вы используете интерпретатор типа *csh* (например *tcsh*) команда с циклом *for* будет выглядеть иначе.
После установки утилит Вы должны создать файл /etc/frad/router.conf Вы можете использовать в качестве образца следующий файл:

```
# /etc/frad/router.conf
# Образец файла конфигурации для сети с ретрансляцией кадров.
# Этот файл содержит все допустимые опции. Файл основан на исходном
# коде DOS-драйверов карты Sangoma S502A.
#
# Символ '#' означает начало комментария до конца строки
# Символы пробела и табуляции игнорируются.
# Неизвестные разделы и опции игнорируются
#

[Devices]
Count=1                 # Количество устройств
Dev_1=sdla0             # имя устройства
#Dev_2=sdla1            # имя устройства

# Общие настройки по умолчанию для всех карт.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Максимальная длина IFrame, по умолчанию 4096
# T391=10               # значение параметра T391  5 - 30, по умолчанию 10
# T392=15               # значение параметра T392 5 - 30, по умолчанию 15
# N391=6                # значение параметра N391 1 - 255, по умолчанию 6
# N392=3                # значение параметра N392 1 - 10, по умолчанию 3
# N393=4                # значение параметра N393 1 - 10, по умолчанию 4

# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511
```

```
#
#
# Настройки отдельных устройств
#
#

#
# Первое устройство -- Sangoma S502E
#
[sdla0]
Type=Sangoma            # Тип устройства. На данный момент поддерживаются
                        # только устройства типа SANGOMA
#
# Эти параметры относятся к типу 'Sangoma'
#
# Модель карты Sangoma - S502A, S502E, S508
Board=S502E
#
# Имя файла с тестовой прошивкой
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# Имя файла с прошивкой FR
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Номер порта
Mem=C8                  # Адрес окна в памяти, A0-EE, в зависимости от карты
IRQ=5                   # номер IRQ, не требуется для S502A
DLCIs=1                 # количество устройств DLCI, подсоединенных к
                        # этой карте
DLCI_1=16               # номер первого DLCI, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Опции данного конкретного устройства
#
# Access=CPE            # CPE или NODE, по умолчанию CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # External или Internal, по умолчанию Internal
# Baud=128              # Скорость подключенного CSU/DSU (baud)
# MTU=2048              # Максимальная длина IFrame, по умолчанию 4096
# T391=10               # значение параметра T391 5 - 30, по умолчанию 10
# T392=15               # значение параметра T392 5 - 30, по умолчанию 15
# N391=6                # значение параметра N391 1 - 255, по умолчанию 6
# N392=3                # значение параметра N392 1 - 10, по умолчанию 3
# N393=4                # значение параметра N393 1 - 10, по умолчанию 4

#
# Настройки другой карты
#
# [sdla1]
# Type=FancyCard        # Тип устройства
# Board=                # Тип карты
# Key=Value             # параметры, специфичные для данного типа карт

#
```

```
# Настройки DLCI по умолчанию
#
CIRfwd=64               # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511

#
# Настройки конкретных DLCI.
# Эти настройки можно опустить. Разделы называются
# [DLCI_D<номер устройства>_<номер_DLCI>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

После того, как вы создали файл `/etc/frad/router.conf` Вам осталось настроить сами устройства. Эта настройка лишь чуть-чуть сложнее настройки обычных сетевых устройств, вам лишь нужно помнить, что устройства FRAD должны запускаться перед устройствами DLCI.

```
#!/bin/sh
# Настройка frad-карт и параметров DLCI
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Активирование устройства FRAD
ifconfig sdla0 up
#
# Настройка интерфейсов DLCI и маршрутизации
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
#
```

```
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

8.8 IPX (AF_IPX)

Протокол IPX наиболее распространен в сетях на основе программного обеспечения Novell Netware(tm). В Линуксе есть поддержка этого протокола, позволяющая Линукс-машине выступать в качестве участника или маршрутизатора в IPX-сети.
Опции компиляции ядра:

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Протокол IPX и файловая система NCPFS подробно рассмотрены в *IPX-HOWTO* <IPX-HOWTO.html>.

8.9 NetRom (AF_NETROM)

Устройствам NetRom ядро присваивает имена 'nr0', 'nr1', и т.д.
Опции компиляции ядра:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Протоколы AX25, Netrom и Rose рассмотрены подробно в *AX25-HOWTO* <AX25-HOWTO.html>. Эти протоколы используются операторами 'Amateur Radio'.
Большая часть работы по реализации этих протоколов в Линуксе была выполнена Джонатаном Нейлором (Jonathon Naylor, jsn@cs.nott.ac.uk).

8.10 Протокол Rose (AF_ROSE)

Устройствам Rose ядро присваивает имена 'rs0', 'rs1' и т.д. Поддержка протокола Rose появилась в версиях ядра 2.1.*.
Опции компиляции ядра:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

Протоколы AX25, Netrom и Rose рассмотрены подробно в *AX25-HOWTO* <AX25-HOWTO.html>. Эти протоколы используются операторами 'Amateur Radio'.
Большая часть работы по реализации этих протоколов в Линуксе была выполнена Джонатаном Нейлором (Jonathon Naylor, jsn@cs.nott.ac.uk).

8.11 SAMBA - поддержка протоколов 'NetBEUI' и 'NetBios'.

SAMBA — реализация протокола SMB (Session Management Block). Samba позволяет машинам с операционными системами фирмы Microsoft и других использовать Ваши диски и принтеры. Установка и настройка Samba детально описаны в *SMB-HOWTO* <SMB-HOWTO.html>.

8.12 Поддержка STRIP (Starmode Radio IP)

Устройствам STRIP ядро присваивает имена 'st0', 'st1' и т.д.
Опции компиляции ядра:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        < > STRIP (Metricom starmode radio IP)
```

STRIP — протокол, специально разработанный для радиомодемов Metricom в рамках проекта *MosquitoNet Project* <http://mosquitonet.Stanford.EDU/mosquitonet.html> Стэнфордского Университета. Web-страница проекта содержит много интересной информации, рекомендуем Вам ее посетить, даже если Вы не интересуетесь этим проектом напрямую.
Радиомодемы Metricom подключаются к последовательному порту, используют технологию широкого спектра и способны передавать данные на скорости около 100 Kb/c. Информацию об этих модемах Вы можете найти на *Web-сервере Metricom* <http://www.metricom.com/>.
В настоящее время стандартные сетевые утилиты не поддерживают драйвер STRIP, поэтому Вы должны использовать специализированные утилиты настройки с сервера *MosquitoNet* <http://mosquitonet.Stanford.EDU/strip.html>.
Они включают модифицированную программу *slattach*, которая переводит последовательное tty-устройсво в режим STRIP. После этого настройте полученное устройство 'st[0-9]' так, как если бы это было ethernet-устройство с одним исключением. STRIP-устройства не поддерживают протокол ARP и Вам прийдется вручную создать arp-записи на всех машинах в Вашей STRIP-сети.

8.13 Сети Token Ring

Устройствам Token ring ядро присваивает имена 'tr0', 'tr1' и т.д. Token Ring — сетевой протокол фирмы IBM, созданный для того, чтобы избегать коллизий. При работе в сети Token Ring в каждый момент времени только одна машина — владеющая специальным 'маркером' может передавать данные. После окончания передачи данных маркер передается следующей станции в сети — по кольцу.
Опции компиляции ядра:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Token Ring driver support
        < > IBM Tropic chipset based adaptor support
```

Настройка сети token ring идентична настройке сети ethernet за исключением имени сетевого устройства.

8.14 X.25

X.25 — протокол с коммутацией пакетов, описанный в C.C.I.T.T. (организация стандартизации, признанная большинством телекоммуникационных компаний мира). Поддержка протоколов X.25 и LAPB появилась в последних версиях ядра 2.1.*.
Джонатан Нэйлор (Jonathon Naylor jsn@cs.nott.ac.uk) руководит разработкой и ведет список рассылки, посвященный протоколу X.25 в Линуксе. Для того, чтобы подписаться отправьте письмо по адресу: majordomo@vger.rutgers.edu с текстом "subscribe linux-x25"в теле письма.
Альфа версии утилит настройки можно получить с *ftp.cs.nott.ac.uk* <ftp://ftp.cs.nott.ac.uk/jsn/>.

8.15 Сетевые карты WaveLan

Картам Wavelan ядро присваивает имена 'eth0', 'eth1' и т.д.
Опции компиляции ядра:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        ....
        <*> WaveLAN support
```

WaveLAN — сетевая радиокарта. Выглядит она почти так же как и ethernet-карта и настраивается очень похожим образом
Информацию о картах Wavelan Вы можете получить с сайта *Wavelan.com* <http://www.wavelan.com/>.

9 Разводка кабелей.

Те из вас, кто не боится брать в руки паяльник, могут попробовать сделать кабели для соединения двух Линукс-машин самостоятельно. Ниже приведены диаграммы объясняющие, как это делается.

9.1 Последовательный нуль-модемный кабель.

Нуль-модемные кабели можно делать по-разному. Многие из них просто меняют местами входы приема и передачи данных, а на остальных входах имитируют 'правильный' сигнал. Этого в принципе достаточно, но означает что Вы должны использовать XON/XOFF-протокол flow control, а это менее эффективно, чем аппаратный flow control. Приведенная диаграмма — наилучшей нуль-модемный кабель, позволяющий использовать аппаратный (RTS/CTS) flow control.

```
Название   Номер                              Номер
Tx Data    2    ------------------------------ 3
Rx Data    3    ------------------------------ 2
RTS        4    ------------------------------ 5
CTS        5    ------------------------------ 4
Ground     7    ------------------------------ 7
DTR        20   -\---------------------------- 8
DSR        6    -/
RLSD/DCD   8    ----------------------------/- 20
                                           \- 6
```

9.2 Кабель для соединения через параллельные порты (PLIP)

Если Вы собираетесь использовать протокол PLIP, приведенный ниже кабель будет работать независимо от типа параллельного порта, установленного в Вашей машине.

```
Название    Номер          Номер
STROBE      1*
D0->ERROR   2  ----------- 15
D1->SLCT    3  ----------- 13
D2->PAPOUT  4  ----------- 12
D3->ACK     5  ----------- 10
D4->BUSY    6  ----------- 11
D5          7*
D6          8*
D7          9*
```

```
ACK->D3      10 ----------- 5
BUSY->D4     11 ----------- 6
PAPOUT->D2   12 ----------- 4
SLCT->D1     13 ----------- 3
FEED         14*
ERROR->D0    15 ----------- 2
INIT         16*
SLCTIN       17*
GROUND       25 ----------- 25
```

Замечания:

- Не подключайте входы, помеченные символом '*'.
- Входы 18,19,20,21,22,23 и 24 заземлены.
- Если у кабеля есть металлическая оплетка, она должна быть подключена **только к одному** из разъемов DB-25.

Внимание! Неверно спаянный кабель может повредить контроллер. Будьте очень внимательны и тщательно проверьте все провода перед тем как использовать кабель.
Вы можете делать PLIP-кабели достаточно длинными, однако не советуем Вам делать их длиннее 1 метра. Будьте особенно внимательны при прокладке кабеля в местах с сильным электромагнитным излучением — около высоковольтных линий, радиопередатчиков и т.п. Если Вы хотите соединить две машины на большом расстоянии друг от друга, советуем Вам установить на них ethernet-карты и провести коаксиальный кабель.

9.3 10base2 (тонкий коаксиальный Ethernet-кабель)

10base2 — стандарт ethernet, использующий 52x-омный коаксиальный кабель диаметром приблизительно 5 мм. При соединении машин тонким коаксиальным ethernet-кабелем нужно помнить следующие правила. Терминаторы должны быть установлены на **обоих** концах кабеля. Терминатор — 52x-омный резистор, благодаря которому сигнал не отражается от концов кабеля. Без терминатора сеть будет работать ненадежно или не будет работать вообще. Для подключения машин используют T-образные соединители (T-коннекторы). Сеть будет выглядеть так:

```
|==========T=============T=============T==========T==========|
           |             |             |          |
           |             |             |          |
         -----         -----         -----      -----
         |   |         |   |         |   |      |   |
         -----         -----         -----      -----
```

где '|' на каждом конце обозначает терминатор, '======' означает коаксиальный кабель, а 'T' обозначает 'T-коннектор'. Кабель от 'T-коннектора' до ethernet-карты должен быть как можно короче, желательно чтобы 'T-коннектор' был непосредственно включен в ethernet-карту.

9.4 Ethernet-кабель 'Витая пара'

Если у Вас только две машины с ethernet-картами под витую пару, то Вам не нужно устанавливать хаб. Вы можете соединить машины напрямую. Диаграмма для кабеля приведена в *Ethernet-HOWTO* <Ethernet-HOWTO.html>

10 Глоссарий.

Ниже приведен список наиболее важных терминов, использующихся в данном документе.

ARP

Сокращение от*Address Resolution Protocol (Протокол преобразования адресов)*. С помощью этого протокола машина преобразует IP-адрес в аппаратный.

ATM

Сокращение от *Asynchronous Transfer Mode (Асинхронный режим передачи)*. Технология ATM — технология с коммутацией пакетов, которая передает данные маленькими пакетами стандартного размера.

клиент

Как правило этот термин обозначает программное обеспечение на машине пользователя, позволяющее ему использовать другие машины в сети. Оконная система X11 — исключение, в ней на пользовательской машине запущен сервер, а клиент работает на удаленной машине. В случае соединения точка-точка (например *slip* или *ppp*) клиентом является сторона, пытающаяся установить соединение, а другая сторона является сервером.

пакет(датаграмма)

Пакет — конечный блок данных, содержащий адреса — основной элемент передачи данных в IP-сетях.

DLCI

Сокращение от Data Link Connection Identifier (Идентификатор подключения к соединению). Используется для того, чтобы обозначить виртуальное соединение точка-точка в сетях с ретрансляцией кадров. Эти идентификаторы обычно присваиваются провайдерами сетей с ретрансляцией кадров.

Ретрансляция кадров

технология, призванная обеспечивать передачу данных с переменной интенсивностью потока. Стоимость работы в таких сетях снижается за счет того, что много пользователей используют общую сеть и расчет делается на то, что они будут пользоваться сетью в разное время.

Аппаратный адрес

Число, которое однозначно определяет машину в сети на аппаратном уровне. Примером могут служить *Ethernet-адреса* и *AX.25-адреса*.

ISDN

Сокращение *Integrated Services Digital Network (Цифровая сеть интегрированных услуг)*. ISDN предоставляет стандартизованные средства передачи данных разных типов. Технически ISDN реализована как сеть с коммутацией соединений.

ISP

Сокращене от Internet Service Provider (Поставщик интернет-услуг). Это организация или компания, предоставляющая возможность подключения к интернет.

IP-адрес

Число, однозначно идентифицирующее машину в TCP/IP-сети. Адрес состоит из 4-х байт и обычно записывается в десятичном виде — каждый байт представлен десятичным числом, байты разделены точками.

MSS

Сокращение от Maximum Segment Size (Максимальный размер сегмента) — максимальный объем данных, которые можно передать за один раз. Если Вы хотите избежать локальной дефрагментации пакетов, MSS следует установить равным MTU-длина IP-заголовка.

MTU

Сокращение от Maximum Transmission Unit (Максимальный блок передачи). Это параметр, который определяет максимальный размер IP-пакета, который можно передавать через IP-интерфейс не разбивая на меньшие части. На разных участках следования IP-пакета MTU может быть разным, то есть даже если пакет не был фрагментирован в локальной сети, он может быть фрагментирован далее по своему маршруту. Типичные значения MTU — 1500 байт для ethernet-сети и 576 байт для SLIP-соединения.

маршрут

Маршрут — путь, который IP-пакет проделывает от точки отправки до точки назначения

сервер

Как правило этим термином обозначают программное обеспечение на удаленной от пользователя машине, которое обрабатывает сетевые запросы и предоставляет пользователю (или пользователям) определенный сервис. Примерами являются *ftp*, *Сетевая файловая система NFS* или *Сервер преобразования имен*. В случае соединений точка-точка под сервером подразумевают машину, к которой пытаются подключится и установить соединение. Вызывающую машину называют клиентом.

окно

Окно — максимальный объем данных, который принимающая сторона может обработать в заданный момент времени.

11 Линукс для интернет-провайдера ?

Если Вас интересует возможность использования Линукса интернет-провайдером, рекомендуем Вам посетить web-страницу *Linux ISP homepage* <http://www.anime.net/linuxisp/>. Эта страница содержит массу полезных ссылок.

12 Благодарности

Хотелось бы выразить признательность следующим людям за их участие в создании этого документа (список не упорядочен): Terry Dawson, Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor, Claes Ensson, Ron Nessim, John Minack, Jean-Pierre Cocatrix, Erez Strauss.

13 Авторские права.