Настольная книга Gentoo Linux x86

Sven Vermeulen *автор*
Roy Marples *автор*
Daniel Robbins *автор*
Chris Houser *автор*
Jerry Alexandratos *автор*
Seemant Kulleen *разработчик Gentoo x86*
Tavis Ormandy *разработчик Gentoo Alpha*
Jason Huebel *разработчик Gentoo AMD64*
Guy Martin *разработчик Gentoo HPPA*
Pieter Van den Abeele *разработчик Gentoo PPC*
Joe Kallar *разработчик Gentoo SPARC*
John P. Davis *редактор*
Pierre-Henri Jondot *редактор*
Eric Stockbridge *редактор*
Rajiv Manglani *редактор*
Jungmin Seo *редактор*
Stoyan Zhekov *редактор*
Jared Hudson *редактор*
Colin Morey *редактор*
Jorge Paulo *редактор*
Carl Anderson *редактор*
Jon Portnoy *редактор*
Zack Gilburd *редактор*
Jack Morgan *редактор*
Benny Chuang *редактор*
Erwin *редактор*
Joshua Kinard *редактор*
Tobias Scherbaum *редактор*
Xavier Neys *редактор*
Grant Goodyear *рецензент*
Gerald J. Normandin Jr. *рецензент*
Donnie Berkholz *рецензент*
Ken Nowack *рецензент*
Lars Weiler *участник*
Антон Битков *переводчик*
Андрей Бородай *переводчик*
Василий Голубев *переводчик*
Алексей Глазунов *переводчик*
Игорь Короть *переводчик*
Сергей Кулешов *переводчик, редактор перевода*
Игорь Наум *переводчик*
Антон Филимонов *переводчик*
Азамат Хакимов *переводчик, редактор перевода*
Эльдар Цраев *переводчик*
Analyzer *переводчик*
Алексей Чумаков *ведущий переводчик, редактор перевода*

Обновлено 17 сентября 2006
Исходный документ обновлен 1 января 2010

Содержание:

порядок ее выполнения.

4. **Подготовка дисков**
   Чтобы установить Gentoo, нужно создать подходящие дисковые разделы. В этой главе описывается, как разбить диск для будущего использования.
5. **Установка установочных файлов Gentoo**
   Установка Gentoo выполняется с помощью архива третьей стадии (stage3). В этой главе мы расскажем, как извлечь файл третьей стадии и настроить Portage.
6. **Установка базовой системы Gentoo**
   После установки и настройки третьей стадии в вашем распоряжении оказывается базовая система Gentoo. Здесь описывается, как этого достичь.
7. **Настройка ядра**
   Ядро Linux — сердце каждого дистрибутива. В этой главе описывается, как его настроить.
8. **Настройка параметров системы**
   Вам потребуется отредактировать несколько важных конфигурационных файлов. Из этой главы вы получите представление об этих файлах и поймете, как с ними обращаться.
9. **Установка нужных системных средств**
   Как отмечено ранее, Gentoo — это богатство выбора. В этой главе мы поможем вам выбрать и установить некоторые важные инструменты.
10. **Настройка начального загрузчика**
    Для архитектуры x86 существует несколько загрузчиков. Каждый из них настраивается по-своему. В этой главе мы шаг за шагом объясним вам порядок настройки начального загрузчика для ваших нужд.
11. **Завершение установки Gentoo**
    Установка почти закончена. Осталось создать одну (или несколько) учетных записей для пользователей вашей системы. Как это делается, описано в данной главе.
12. **Чем заняться дальше?**
    Теперь у вас появилась собственная система Gentoo. Но чем же заняться дальше?

- **Работа с Gentoo**
  Здесь вы научитесь работать с Gentoo: устанавливать программное обеспечение, изменять значения переменных, управлять поведением Portage и т.д.
  1. **Введение в Portage**
     В этой главе описываются «простые» шаги, которые вам точно потребуется знать для поддержания в порядке и обслуживания программного обеспечения.
  2. **USE-флаги**
     USE-флаги (признаки использования) — очень важный аспект Gentoo. Прочитав эту главу, вы научитесь обращаться с ними и разберетесь, как USE-флаги влияют на систему.
  3. **Возможности Portage**
     Откройте для себя дополнительные возможности Portage: поддержку распределенной компиляции, кэш компилятора и др.
  4. **Сценарии инициализации**
     В Gentoo используется специальный формат сценариев инициализации (initscript), в котором, например, предусмотрены решения, управляемые зависимостями, и виртуальные сценарии. Здесь описываются эти аспекты, и объясняется, как обращаться со сценариями.
  5. **Переменные среды**
     В Gentoo можно довольно легко управлять системными переменными среды. В этой главе обьясняется, как это делать, и описываются часто используемые переменные.
- **Работа с Portage**
  В этой части подробно описывается Portage, средство управления программным обеспечением Gentoo.
  1. **Файлы и каталоги**
     Чтобы поближе познакомиться с Portage, вам потребуется узнать, где же хранятся его файлы и данные.
  2. **Настройка с помощью переменных**
     Portage полностью настраивается с помощью различных переменных, которые устанавливаются в конфигурационном файле или как переменные среды.
  3. **Смешение ветвей программного обеспечения**
     Программное обеспечение в составе Gentoo подразделяется на ветви в зависимости от стабильности и поддержки различных архитектур. В этой главе рассказывается, как настраивать использование ветвей, а также как при необходимости преодолевать такое разделение.
  4. **Дополнительные средства Portage**

В состав Portage входит несколько дополнительных инструментов, которые могут значительно улучшить ваше впечатление от Gentoo. В этой главе раскрываются секреты использования dispatch-conf и других инструментов.

5. **Отступление от официального дерева**
   Здесь даются советы и показываются приемы «выращивания» собственного дерева Portage, рассказывается о том, как синхронизировать только нужные категории, добавлять свои пакеты и т.д.
6. **Использование ebuild**
   В этой главе рассказывается о шагах, выполняемых Portage при установке программного обеспечения, и об их самостоятельном выполнении с помощью утилиты ebuild.

- **Настройка сети в Gentoo**
  Полное руководство по сетям в Gentoo.
  1. **Начальная настройка**
     Руководство по быстрому запуску и подключению сетевого интерфейса в наиболее распространенных случаях.
  2. **Расширенная настройка**
     Здесь вы изучите, как работает конфигурация — это нужно сделать перед изучением модульного построения сети.
  3. **Модульное построение сети**
     В Gentoo предусмотрены гибкие сетевые средства: здесь рассказывается о выборе различных клиентов DHCP, настройке объединения, образования мостов, виртуальных сетей (VLAN) и т.п.
  4. **Беспроводная сеть**
     Настроить беспроводную сеть не совсем просто. Надеемся, мы поможем вам заставить ее работать.
  5. **Дополнительные возможности**
     Если вы ищете приключений, можете подключить к сети свои собственные функции.
  6. **Управление сетью**
     Для пользователей ноутбуков и тех, кто перемещает свои компьтеры из сети в сеть.

A. Установка Gentoo

1. Об установке Gentoo Linux

1.a. Введение

Добро пожаловать!

Прежде всего, *добро пожаловать* в Gentoo! Вы — на пороге мира больших возможностей и высокой производительности! Gentoo весь пропитан идеей свободы выбора. При установке Gentoo вы не раз убедитесь в этом: вам предстоит выбрать объем самостоятельной компиляции, способ установки Gentoo, службу журналирования системы и т.д.

Gentoo — быстрый современный метадистрибутив, обладающий большой чистотой и гибкостью. Gentoo основан на свободном программном обеспечении, и не скрывает от пользователя, «что под капотом». Portage, система управления пакетами Gentoo, написана на языке Python, что позволяет легко просматривать и изменять ее исходный код. Сборка Gentoo также выполняется из исходных текстов (хотя есть и поддержка бинарных пакетов), а настройка Gentoo выполняется с помощью обычных текстовых файлов. Другими словами — сплошная открытость и свобода!

Важно понимать, что Gentoo развивается именно благодаря *свободе выбора*. Мы стараемся ничего вам не навязывать. А если вам покажется обратное — пожалуйста, сообщите нам об этой оплошности (англ.).

Как организована установка?

Установка Gentoo рассматривается как последовательность из 10 шагов, которым соответствуют главы 2—11. Каждый шаг приводит к определенному состоянию:

- результат шага 1: вы — в рабочей среде, готовой к установке Gentoo

- результат шага 2: ваше подключение к интернету готово для установки Gentoo
- результат шага 3: ваши жесткие диски готовы стать родным домом для Gentoo
- результат шага 4: подготовлена установочная среда, и вы готовы переключиться (chroot) в новую среду
- результат шага 5: развернуты пакеты ядра, общие для всех систем Gentoo
- результат шага 6: вы скомпилировали собственное ядро Linux
- результат шага 7: вы написали большую часть конфигурационных файлов системы Gentoo
- результат шага 8: установлены необходимые системные средства (которые вы сами выбрали из славного списка)
- результат шага 9: установлен и настроен избранный загрузчик ОС, и вы вошли в новую систему Gentoo
- результат шага 10: вы можете начинать изучение своей собственной среды Gentoo Linux!

Мы приложили все усилия, чтобы объяснить вам все «за» и «против» каждого варианта, когда вам предоставляется возможность выбора. При этом один из вариантов помечен как «По умолчанию: ». Другие варианты помечены как «Альтернатива: ». *Не подумайте*, что вариант по умолчанию — наша рекомендация. Нам просто кажется, что именно его выбирает большинство пользователей.

Иногда есть возможность выполнить необязательный шаг. Такие шаги помечены как «Дополнительно: », и не требуются для установки Gentoo. Однако, некоторые из них будут обусловлены вашими предшествующими решениями. Мы будем сообщать об этом, как в момент выбора, так и непосредственно перед описанием необязательных шагов.

Какие варианты установки существуют?

Gentoo можно установить разными способами. Можно скачать и запустить один из установочных компакт-дисков, установить с имеющегося дистрибутива, с загрузочного CD (например, Knoppix), из сетевой загрузочной среды, с дискеты аварийного восстановления и т.д.

В этом руководстве описывается установка с установочных дисков Gentoo, и, в некоторых случаях, с помощью сетевой загрузки (netboot). Предполагается, что вы собираетесь устанавливать самые свежие версии пакетов. Если вам нужна установка, при которой не требуется использование сети, обратитесь к настольным книгам Gentoo 2006.1 (англ.), где даются указания по установке в бессетевой среде.

Если вы планируете использовать GRP (Gentoo Reference Platform — эталонная платформа Gentoo, набор бинарных пакетов, предназначенных для немедленного использования сразу после установки Gentoo), вам *необходимо* следовать инструкциям, приведенных в настольной книге Gentoo 2006.1 (англ.).

Чтобы получить сведения о других способах установки, прочитайте описание альтернативных способов установки. Также рекомендуется прочитать полезные советы по установке Gentoo. Если вы почувствуете, что приведенные указания по установке слишком подробны, обратитесь к краткому руководству по установке (см. перечень документации), если, конечно, такое существует для вашей архитектуры.

Кроме того, вы можете выбрать между компиляцией своей системы полностью «с нуля» или установкой заранее собранной среды, позволяющей запустить Gentoo практически моментально. Естественно, есть и промежуточные варианты, в которых вы не компилируете все подряд, а начинаете с полуготовой системы.

Появились затруднения?

Если при установке вы столкнулись с проблемой (или с ошибкой в документации по установке), войдите в нашу систему распределения запросов (англ.) и убедитесь, что такая ошибка еще не заявлена. В этом случае, создайте отчет об ошибке, чтобы мы о ней позаботились. Не бойтесь разработчиков, которым выпадает работа над (вашими) ошибками — людей они обычно не едят.

Обратите внимание, что, хотя руководство, которое вы сейчас читаете, посвящено определенной архитектуре, в нем упоминаются и другие архитектуры. Это связано с тем, что значительная часть текста настольной книги Gentoo является общей для всех архитектур (чтобы не дублировать работу, а также из-за острой нехватки разработчиков). Во избежание путаницы, мы стараемся сократить такие упоминания до минимума.

Если вы не уверены, пользовательская ли у вас ошибка (какую-то ошибку совершили вы, хотя внимательно

прочитали документацию), или программная (какую-то ошибку совершили мы, хотя тщательно тестировали установку/документацию), то не стесняясь, заходите на канал #gentoo сервера irc.freenode.net. Разумеется, мы в любом случае будем вам рады :)

Если у вас есть вопрос, касающийся Gentoo, сначала загляните в список распространенных вопросов (FAQ), входящий в состав документации Gentoo. Можно также обратиться к FAQ (англ.) в наших форумах. Если вы не найдете там ответа, задайте вопрос на #gentoo, нашем IRC-канале на irc.freenode.net. Да-да, кое-кто из нас — маньяки, висящие на IRC :-)

2. Выбор подходящего источника для установки

2.a. Аппаратные требования

Введение

Прежде чем начать, перечислим аппаратные требования, необходимые для успешной установки Gentoo на вашем компьютере.

Аппаратные требования

Центральный процессор	**i486 или новее**
Память	**64 МБ**
Дисковое пространство	**1.5 ГБ (исключая пространство подкачки)**
Пространство подкачки	**не менее 256 МБ**

2.b. Установочные компакт-диски Gentoo

Введение

Установочные компакт-диски Gentoo — это загрузочные диски, содержащие самодостаточную среду Gentoo. Они позволяют загружать Linux прямо с компакт-диска. При запуске определяются ваши устройства и загружаются соответствующие драйверы. Эти диски сопровождаются разработчиками Gentoo.

Все установочные компакт-диски позволяют загрузиться, настроить сеть, разметить разделы жесткого диска и начать установку Gentoo из интернета. В настоящее время мы выпускаем два установочных компакт-диска, одинаково подходящих для установки Gentoo с загрузкой последних версий существующих пакетов из интернета.

Если вы хотите установить Gentoo без работоспособного соединения с интернетом, или желаете использовать одну из существующих программ установки, пожалуйста, следуйте указаниям по установке, приведенным в настольной книге Gentoo 2006.0 (англ.).

В настоящее время мы выпускаем два установочных компакт-диска:

- *Минимальный* установочный диск Gentoo, маленький загрузочный компакт-диск без излишеств, единственное назначение которого — загрузить систему, подготовить подключение к сети и перейти к установке Gentoo.
- *Живой установочный диск* Gentoo, содержащий все необходимое для установки Gentoo. На нем есть графическая среда, графическая и консольная программы автоматической установки, и, конечно, указания по установке для вашей архитектуры.

Чтобы помочь вам решить, какой из них нужен, мы привели главные преимущества и недостатки каждого диска.

Минимальный установочный диск Gentoo

Минимальный установочный диск называется `install-x86-minimal-2006.0.iso` и занимает всего 49 МБ. Gentoo можно устанавливать с этого диска только при постоянном наличии работоспособного

подключения к интернету.

Минимальный установочный диск	За и против
+	**наименьший объем загрузки из сети**
–	**не содержит ни архива stage3, ни снимков Portage, ни готовых двоичных пакетов, и поэтому не подходит для установки без сети**

Живой установочный диск Gentoo

Живой установочный диск называется `install-x86-universal-2006.0.iso` и занимает 697 МБ. С этого диска вы сможете установить Gentoo даже без исправного подключения к интернету, на случай, если в процессе установки Gentoo на свой компьютер у вас появится неодолимое желание занести его еще и на соседний :)

Живой установочный диск Gentoo	За и против
+	**содержит все, что требуется; установку можно выполнять даже без сетевого подключения**
–	**огромный объем загрузки**

Файл Stage3

Архив третьей стадии — stage3 — это файл, содержащий минимальную среду Gentoo, пригодную для продолжения установки Gentoo в соответствии с инструкциями, данными в этом руководстве. Когда-то в настольной книге Gentoo описывались варианты установки с использованием файла любой из трех существующих стадий создания системы. Несмотря на то, что в Gentoo до сих представлены файлы stage1 и stage2, в официальном способе установки используется архив stage3. Если вас интересует установка Gentoo из файлов stage1 или stage2, пожалуйста, обратитесь к сборнику распространенных вопросов по Gentoo, раздел как установить Gentoo, используя файлы Stage1 или Stage2?

Архивы stage3 можно загрузить из `releases/x86/2006.0/stages/` или с любого из официальных зеркал Gentoo; на «живом диске» они не поставляются.

2.c. Скачивание, запись и загрузка с установочного диска Gentoo

Скачивание образа и запись на диск

Итак, вы выбрали нужный установочный диск Gentoo. Начнем с загрузки его образа из сети, и его записи на компакт-диск. Ранее обсуждались варианты доступных дисков, но где же их взять?

Любой из установочных дисков (а при желании — заодно и диск пакетов, если есть) можно загрузить с одного из наших зеркал. Установочные диски расположены в каталоге `releases/x86/2006.0/installcd`.

В этом каталоге находятся ISO-файлы. Это полные образы компакт-дисков, которые можно записать на CD-R.

Если вас волнует, не искажен ли загруженный файл, можно сверить его контрольную сумму MD5 с рассчитанной нами (например, `install-x86-minimal-2006.0.iso.DIGESTS`). Вычислить контрольную сумму MD5 можно утилитой `md5sum` в Linux/Unix, или программой md5sum (англ.) для Windows.

Другой способ убедиться в пригодности загруженного файла — с помощью GnuPG сверить криптографическую подпись с той, что хранится у нас (в файле с расширением `.asc`). Скачайте файл подписи, затем получите открытый ключ:

Листинг 3.1: Получение открытого ключа

```
$ gpg --keyserver subkeys.pgp.net --recv-keys 17072058
```

Теперь сверьте подпись:

Листинг 3.2: Сверка криптографической подписи

```
$ gpg --verify <файл подписи> <скачанный образ диска>
```

Записывать загруженный ISO-образ на компакт-диск нужно в «сыром» режиме. Как это сделать, сильно зависит от используемой программы. Здесь мы обсудим запись с помощью `cdrecord` и `K3B`. Дополнительные сведения находятся в нашем сборнике распространенных вопросов.

- При использовании cdrecord, просто введите `cdrecord dev=/dev/hdc <скачанный iso-файл>` (`/dev/hdc` замените на путь к вашему устройству CD-RW).
- При использования K3B, выберите `Tools` > `CD` > `Burn Image` (`Инструменты` > `Компакт-диск` > `Прожечь образ`). Затем в области 'Image to Burn' ('Образ для записи') укажите свой ISO-файл. В завершение нажмите `Start` (`Запуск`).

Загрузка с установочного компакт-диска

Важно: Перед тем, как приступить, полностью прочитайте этот подраздел, так как перед последующими действиями вам, скорее всего, больше не представится такая возможность.

Записав установочный компакт-диск, пора его загрузить. Уберите все компакт-диски из своих приводов CD, перезагрузите компьютер и войдите в BIOS. В зависимости от BIOS, для этого обычно нужно нажать DEL, F1 или ESC. В BIOS измените порядок загрузки так, чтобы обращение к CD-ROM выполнялось до обращения к жесткому диску. Этот параметр часто задается в разделе «CMOS Setup». Если порядок загрузки не изменить, система просто перезагрузится с жесткого диска, игнорируя CD-ROM.

Теперь поместите установочный диск в привод CD-ROM (наконец-то!) и перезагрузитесь. Должно появиться загрузочное приглашение. Здесь можно просто нажать ENTER, чтобы запустить процесс загрузки с параметрами по умолчанию, или загрузить установочный диск по-своему: указать ядро, потом загрузочные параметры, а затем нажать ENTER.

Указать ядро? Да, на нашем установочном компакт-диске есть несколько вариантов ядра. По умолчанию используется `gentoo`. Другие ядра предназначены для специфических аппаратных задач; вариант `-nofb` отключает кадровый буфер.

Ниже перечислены доступные варианты ядра:

Ядро	Описание
gentoo	**ядро 2.6 с многопроцессорной поддержкой (используется по умолчанию)**
gentoo-nofb	**аналогично `gentoo`, но без поддержки кадрового буфера**
memtest86	**для выявления ошибок локальной оперативной памяти**

Также можно указать параметры ядра, явно включающие или отключающие определенные режимы. Приведенный список соответствует тому, что выводится при нажатия F2 в экране загрузки.

Листинг 3.3: Параметры, которые можно сообщить выбранному ядру

```
- agpgart       загрузка agpgart (используйте при сбоях графики, зависаниях)
- acpi=on       включение поддержки ACPI
- ide=nodma     принудительное отключение DMA для сбойных IDE-устройств
- doscsi        поиск scsi-устройств (нарушает работу некоторых ethernet-плат)
- dopcmcia      запуск службы pcmcia для PCMCIA-приводов компакт-дисков
- nofirewire    отключение в initrd модулей firewire (предназначенных для
                приводов компакт-дисков с подключением firewire и т.п.)
- nokeymap      отключение выбора неамериканских раскладок клавиатуры
- docache       кэширование исполняемой части компакт-диска в памяти, позволяет
                размонтировать /mnt/cdrom для установки другого диска
- nodetect      запрет запуска hwsetup/kudzu и hotplug
- nousb         отключение загрузки модулей usb в initrd, отключает hotplug
- nodhcp        отмена автоматического запуска dhcp при обнаружении сетевой
                платы
- nohotplug     отключение загрузки службы hotplug
- noapic        отключение apic (попробуйте, если есть аппаратные проблемы
                с устройствами scsi, сетевыми платами и т.д.)
```

```
 - noevms        отключение поддержки модулей EVMS2
 - nolvm2        отключение поддержки модулей LVM2
 - hdx=stroke    позволяет использовать жесткий диск целиком, даже если BIOS
                 не поддерживает диски большого размера
 - noload=module1,[module2,[...]]
                 запрет загрузки определенных модулей ядра
```

Теперь загрузите систему с компакт-диска, выбрав ядро (если вас не устраивает ядро по умолчанию `gentoo`) и указав дополнительные параметры. В качестве примера мы покажем, как загрузить ядро `gentoo` с параметром `dopcmcia`:

Листинг 3.4: Загрузка установочного диска

```
boot: gentoo dopcmcia
```

Появится заставка с полосой индикатора загрузки. Если вы устанавливаете Gentoo на машину с неамериканской раскладкой клавиатуры, нужно немедленно нажать ALT+F1, чтобы переключиться в подробный режим, и следовать появившимся указаниям. Если ничего не выбрано в течение 10 секунд, устанавливается раскладка по умолчанию (клавиатура США), а загрузка продолжается. По окончании загрузки для вас будет выполнен автоматический вход в «живой» Gentoo Linux в качестве «root» (администратора). В текущей консоли должно появиться прилашение root («#»). Также можно переключаться в другие консоли, нажимая ALT-F2, ALT-F3 и ALT-F4. Вернуться в первоначальную консоль можно нажатием ALT-F1.

Приступим к дополнительной настройке оборудования.

Дополнительная настройка оборудования

При загрузке с установочного компакт-диска система пытается определить все устройства и загрузить соответствующие модули для их поддержки. В подавляющем большинстве случаев она очень хорошо справляется с этим. Однако, в редких случаях некоторые из нужных модулей не загружаются автоматически. Если какие-либо устройства пропущены при автоматическом опросе шины PCI, модули ядра для их поддержки вам придется загрузить вручную.

В следующем примере мы попробуем загрузить модуль `8139too` (для поддержки некоторых типов сетевого интерфейса):

Листинг 3.5: Загрузка модуля ядра

```
# modprobe 8139too
```

Если вам нужна поддержка PCMCIA, требуется запустить сценарий инициализации `pcmcia`:

Листинг 3.6: Запуск сценария инициализации PCMCIA

```
# /etc/init.d/pcmcia start
```

Дополнительно: оптимизация скорости жесткого диска

Если вы опытный пользователь, возможно, вам захочется настроить скоростные параметры жесткого диска IDE с помощью программы `hdparm`. Указав параметр `-tT`, можно измерить скорость вашего диска (для уточнения значений запускайте программу несколько раз подряд):

Листинг 3.7: Тестирование производительности диска

```
# hdparm -tT /dev/hda
```

Для настройки вы можете использовать любой из приведенных примеров (или поэкспериментировать самостоятельно), при необходимости указав название своего диска вместо `/dev/hda`:

Листинг 3.8: Оптимизация скорости жесткого диска

```
Включение DMA:                  # hdparm -d 1 /dev/hda
Установка безопасных значений:  # hdparm -d 1 -A 1 -m 16 -u 1 -a 64 /dev/hda
```

Дополнительно: учетные записи пользователей

Если вы собираетесь дать другим людям доступ к своей системе, или намерены входить в чат `irssi` без прав администратора (из соображений безопасности), потребуется создать учетные записи пользователей и изменить пароль администратора.

Для изменения пароля администратора используйте `passwd`:

Листинг 3.9: Изменение пароля администратора

```
# passwd
New password: (введите новый пароль)
Re-enter password: (введите новый пароль еще раз)
```

Для создания учетной записи пользователя сначала задаются его реквизиты, а затем вводится пароль. Для этого служат программы `useradd` и `passwd`. В следующем примере мы создадим пользователя «john»:

Листинг 3.10: Создание учетной записи пользователя

```
# useradd -m -G users john
# passwd john
New password: (введите новый пароль для john)
Re-enter password: (введите новый пароль Джона еще раз)
```

Вы также можете изменить свой идентификатор пользователя с администратора (root) на вновь созданного пользователя командой `su`:

Листинг 3.11: Изменение идентификатора пользователя

```
# su - john
```

Дополнительно: просмотр документации во время установки

Если вам нужна возможность просмотра настольной книги Gentoo в процессе установки (не важно, из сети или с компакт-диска), убедитесь, что вы создали учетную запись обычного пользователя (см. дополнительно: учетные записи пользователей). Затем нажмите `ALT+F2` для перехода в другой терминал, и войдите в систему.

Для чтения документации с компакт-диска, вы можете сразу запустить `links`:

Листинг 3.12: Просмотр документации с компакт-диска

```
# links /mnt/cdrom/docs/html/index.html
```

Но лучше открывать настольную книгу Gentoo из сети (там она новее, чем на компакт-диске). Для этого тоже можно пользоваться `links`, но только после *настройки сети* (иначе вы не сможете выйти в интернет, чтобы прочесть документ):

Листинг 3.13: Просмотр документации, находящейся в сети

```
# links http://www.gentoo.org/doc/ru/handbook/handbook-x86.xml
```

На исходный терминал можно переключаться нажатием `ALT+F1`.

Дополнительно: запуск демона SSH

Если вы хотите разрешить другим пользователям доступ к своей системе в процессе установки Gentoo (возможно затем, чтобы они помогли вам с установкой или даже провели ее за вас), для них потребуется создать учетные записи, а возможно, даже дать им пароль администратора (делайте это *только* в том случае, *если* вы им **полностью доверяете**).

Чтобы запустить демон SSH, выполните следующую команду:

Листинг 3.14: Запуск демона SSH

```
# /etc/init.d/sshd start
```

Для использования sshd, сначала требуется настроить сеть. Переходите к главе настройка сети.

3. Настройка сети

3.a. Автоматическое подключение к сети

Может, она уже настроена?

Если ваша система подсоединена к сети Ethernet, в которой есть сервер DHCP, очень вероятно, что сетевое подключение на вашей машине уже автоматически настроено. Если так, вы сможете воспользоваться многими сетевыми командами, находящимися на установочном компакт-диске, например, `ssh`, `scp`, `ping`, `irssi`, `wget` и `links`.

Если сеть уже настроена, команда `/sbin/ifconfig` должна показывать сетевые интерфейсы кроме lo, например, eth0:

Листинг 1.1: /sbin/ifconfig для рабочей сетевой конфигурации

```
# /sbin/ifconfig
(...)
eth0      Link encap:Ethernet  HWaddr 00:50:BA:8F:61:7A
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::50:ba8f:617a/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1498792 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1284980 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1984 txqueuelen:100
          RX bytes:485691215 (463.1 Mb)  TX bytes:123951388 (118.2 Mb)
          Interrupt:11 Base address:0xe800
```

Дополнительно: указание прокси-серверов

Если вы подключены к интернету через прокси-сервер, при установке вам может потребоваться ввести сведения о нем. Задать прокси-сервер очень просто: нужно определить переменные, содержащие необходимые сведения.

В большинстве случаев в определении переменных достаточно указать имя прокси-сервера. Предположим, например, что прокси-сервер называется `proxy.gentoo.org`, а его порт — `8080`:

Листинг 1.2: Указание прокси-сервера

```
 (если прокси-сервер фильтрует трафик HTTP)
# export http_proxy="http://proxy.gentoo.org:8080"
 (если прокси-сервер фильтрует трафик FTP)
# export ftp_proxy="ftp://proxy.gentoo.org:8080"
 (если прокси-сервер фильтрует трафик RSYNC)
# export RSYNC_PROXY="proxy.gentoo.org:8080"
```

Если прокси-сервер запрашивает имя пользователя и пароль, для значения переменной следует использовать такой формат:

Листинг 1.3: Указание имени/пароля в адресе прокси-сервера

```
http://имя_пользователя:пароль@proxy.gentoo.org:8080
```

Проверка сети

Может оказаться полезным проверить отклик DNS-сервера вашего поставщика услуг интернета (адрес сервера находится в `/etc/resolv.conf`), а также произвольного веб-сайта, чтобы убедиться, что ваши пакеты выходят в интернет, разрешение имен DNS работает и т.д.

Листинг 1.4: Проверка доступности сети

```
# ping -c 3 www.yahoo.com
```

Сеть доступна? Тогда пропустите остаток этого раздела, и переходите к разделу подготовка дисков. Если сеть все же недоступна, то, к сожалению, вам придется еще поработать над ее настройкой.

3.b. Автоматизированная настройка сети

Если сеть не заработает сразу, при установке с некоторых носителей можно использовать `net-setup` (для обычных или беспроводных сетей), `pppoe-setup` (для пользователей ADSL) или `pptp` (для пользователей PPTP; есть на x86, amd64, alpha, ppc и ppc64).

Если на вашем установочном диске нет ни одного из этих средств, или сеть еще не подключена, приступайте к ручной настройке сети:

- пользователи обычной сети Ethernet — переходите к разделу по умолчанию: использование net-setup
- пользователи ADSL — переходите к разделу альтернатива: использование RP-PPPoE
- пользователи PPTP — переходите к разделу альтернатива: использование PPTP

По умолчанию: использование net-setup

Простейший способ настройки сети, если она не произошла автоматически — запуск сценария `net-setup`:

Листинг 2.1: Запуск сценария net-setup

```
# net-setup eth0
```

`net-setup` задаст вам несколько вопросов о вашей сетевой среде. В результате его работы у вас должно появиться работоспособное подключение к сети. Проверьте сетевое подключение, как это описано выше. Если проверка прошла успешно, примите наши поздравления — теперь вы готовы к установке Gentoo. Пропустите оставшуюся часть этого раздела и приступайте к подготовке дисков.

Если ваша сеть все еще не заработала, переходите к ручной настройке сети.

Альтернатива: использование RP-PPPoE

Для простоты подключения к интернету по PPPoE, в установочный диск (любой версии) включен `rp-pppoe`. Для настройки соединения используйте сценарий `pppoe-setup`, входящий в комплект. У вас будет запрошена информация о сетевом устройстве ethernet, подключенном к adsl-модему, имя пользователя, пароль, IP-адреса ваших серверов DNS. Также задается вопрос, нужно ли включать базовый межсетевой экран (firewall).

Листинг 2.2: Использование rp-pppoe

```
# pppoe-setup
# pppoe-start
```

Если что-то пошло не так, проверьте, правильно ли вы ввели имя пользователя и пароль, посмотрев в `/etc/ppp/pap-secrets` или `/etc/ppp/chap-secrets`, и убедитесь, что устройство ethernet указано

верно. Если ваше устройство не видно в системе, потребуется загрузить соответствующие сетевые модули. Для этого нужно перейти к разделу ручная настройка сети где мы объясняем, как из загрузить.

Если же все заработало, переходите к подготовке дисков.

Альтернатива: использование PPTP

Примечание: Поддержка PPTP имеется только для архитектуры x86.

Если вам нужна поддержка PPTP, можно использовать `pptpclient`, который входит в состав нашего установочного диска. Но сначала нужно обеспечить правильность настройки. Отредактируйте файлы `/etc/ppp/pap-secrets` или `/etc/ppp/chap-secrets` так, чтобы в них находилось правильное сочетание имени пользователя и пароля.

Листинг 2.3: Редактирование /etc/ppp/chap-secrets

```
# nano -w /etc/ppp/chap-secrets
```

Затем, если нужно, измените параметры PPTP в файле `/etc/ppp/options.pptp`:

Листинг 2.4: Редактирование /etc/ppp/options.pptp

```
# nano -w /etc/ppp/options.pptp
```

Когда все будет готово, просто запустите `pptp` (с параметрами, которые вы не стали прописывать в `options.pptp`), чтобы соединиться с сервером:

Листинг 2.5: Подключение к серверу коммутируемого доступа

```
# pptp <server ip>
```

Теперь переходите к подготовке дисков.

3.c. Ручная настройка сети

Загрузка нужных сетевых модулей

При загрузке установочный диск пытается выявить все установленные устройства и загружает подходящие модули ядра (драйверы) для поддержки вашего оборудования. В подавляющем большинстве случаев он очень хорошо справляется с этой работой. Однако, в некоторых случаях он может не справиться с автозагрузкой нужных вам модулей ядра.

Если `net-setup` или `pppoe-setup` не удалось загрузить нужный модуль, возможно, ваша сетевая плата сразу не обнаружена. При этом вам может потребоваться ручная загрузка необходимых модулей ядра.

Чтобы выяснить, какие модули ядра для поддержки сети существуют, используйте `ls`:

Листинг 3.1: Поиск имеющихся модулей

```
# ls /lib/modules/`uname -r`/kernel/drivers/net
```

Если вы найдете драйвер для своей сетевой платы, для загрузки модуля ядра используйте `modprobe`:

Листинг 3.2: Использование modprobe для загрузки модуля ядра

```
(для примера загрузим модуль pcnet32)
# modprobe pcnet32
```

Чтобы убедиться, что ваша сетевая плата теперь обнаружена, используйте `ifconfig`. Если сетевая плата

обнаружена, результат выглядит подобным образом:

Листинг 3.3: Проверка доступности сетевой платы (удачная)

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FE:FD:00:00:00:00
          BROADCAST NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Однако, если вы получите такое сообщение об ошибке, сетевая плата не обнаружена:

Листинг 3.4: Проверка доступности сетевой платы (неудачная)

```
# ifconfig eth0
eth0: error fetching interface information: Device not found
```

Если в вашей системе установлено несколько сетевых плат, они будут называться *eth0*, *eth1* и т.д. Убедитесь, что сетевая плата, которую вы собираетесь использовать, работает хорошо, и в дальнейшем не забудьте везде подставлять верное имя. Мы предполагаем, что используется сетевая плата *eth0*.

Когда ваша сетевая плата обнаружена, можно попробовать снова запустить `net-setup` или `pppoe-setup` (которые теперь должны сработать), но на случай, если вы из крутых, мы опишем, как настроить сеть вручную.

Выберите один из следующих разделов, в зависимости от необходимого вида настройки:

- использование DHCP для автоматического присвоения IP-адреса
- подготовка беспроводного доступа, если у вас есть беспроводная плата
- освоение сетевой терминологии: рассказ о том, что нужно знать о подключении к сети
- использование ifconfig и route: описание ручной настройки сети

Использование DHCP

DHCP (Dynamic Host Configuration Protocol — протокол динамической настройки хоста) дает возможность автоматически получить параметры сетевого подключения (IP-адрес, маску сети, широковещательный адрес, шлюз, сервера имен и т. д.) Все это работает, только когда в вашей сети есть сервер DHCP (или ваш поставщик предоставляет услугу DHCP). Чтобы сетевой интерфейс получал эти сведения автоматически, используйте `dhcpcd`:

Листинг 3.5: Использование dhcpcd

```
# dhcpcd eth0
Некоторые сетевые администраторы требуют, чтобы вы использовали
имя хоста и домена, назначенное сервером DHCP.
В этом случае используйте
# dhcpcd -HD eth0
```

Если это сработало (попробуйте «попинговать» какой-нибудь сервер интернета, например, Google), то у вас все настроено, и можно двигаться дальше. Пропустите остаток этого раздела и приступайте к подготовке дисков.

Подготовка беспроводного доступа

Примечание: Поддержка команды `iwconfig` есть только на установочных дисках для платформ x86, amd64 и ppc. В других случаях вы все же сможете запустить расширения, следуя инструкциям проекта linux-wlan-ng (англ.).

Если вы используете беспроводную плату (802.11), перед дальнейшими действиями может потребоваться настройка параметров беспроводного подключения. Для просмотра текущей настройки беспроводной платы можете использовать `iwconfig`. При запуске `iwconfig` вы увидите подобные сведения:

Листинг 3.6: Вывод текущих параметров беспроводного подключения

```
# iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"GentooNode"
          Mode:Managed  Frequency:2.442GHz  Access Point: 00:09:5B:11:CC:F2
          Bit Rate:11Mb/s   Tx-Power=20 dBm   Sensitivity=0/65535
          Retry limit:16   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:25/10  Signal level:-51 dBm  Noise level:-102 dBm
          Rx invalid nwid:5901 Rx invalid crypt:0 Rx invalid frag:0 Tx
          excessive retries:237 Invalid misc:350282 Missed beacon:84
```

Примечание: Некоторые беспроводные платы могут называться подобно `wlan0` или `ra0`, а не `eth0`. Для определения верного имени устройства запускайте `iwconfig` без каких-либо параметров командной строки.

Для большинства пользователей может потребоваться изменение только двух параметров: ESSID (названия беспроводной сети) и ключа шифрования WEP. Если ESSID и указанный адрес точки доступа уже принадлежат вашей точке доступа, и вы не используете WEP, значит, ваше беспроводное подключение работает. Если вам необходимо изменить свой ESSID или добавить ключ WEP, можно запустить следующие команды:

Листинг 3.7: Замена ESSID и/или добавление ключа WEP

```
(так название сети устанавливается в "GentooNode")
# iwconfig eth0 essid GentooNode

(так устанавливается шестнадцатиричный ключ WEP)
# iwconfig eth0 key 1234123412341234abcd

(так устанавливается текстовый ключ (ASCII); вначале нужно добавлять "s:")
# iwconfig eth0 key s:some-password
```

Проверить сделанную настройку можно, снова запустив `iwconfig`. Как только ваша беспроводная сеть заработает, вы можете перейти к установке параметров сети, относящихся к протоколу IP, которые описываются в следующем разделе (освоение сетевой терминологии), или использовать утилиту `net-setup`, как описано ранее.

Освоение сетевой терминологии

Примечание: Если вы знаете свой IP-адрес, широковещательный адрес, маску сети и серверы имен, можете пропустить этот подраздел и перейти к разделу использование ifconfig и route.

Если все, рассказанное выше, не помогло, вам придется настроить свою сеть вручную. Это совсем нетрудно. Однако, вам понадобится освоить кое-какую сетевую терминологию, знание которой требуется для удовлетворительной настройки сети. Прочитав этот текст, вы узнаете, что такое *шлюз*, зачем служит *маска сети*, как формируется *широковещательный адрес*, и зачем нужны *серверы имен*.

В сети узлы (хосты, компьютеры) различаются по *IP-адресу* (адресу протокола интернета). Такой адрес — это сочетание четырех чисел от 0 до 255. Ну, по крайней мере, так мы его воспринимаем. В действительности, IP-адрес состоит из 32 бит (единиц и нулей). Давайте рассмотрим пример:

Листинг 3.8: Пример IP-адреса

```
IP-адрес (числа):       192.168.0.2
IP-адрес (биты):        11000000 10101000 00000000 00000010
                        -------- -------- -------- --------
                           192      168       0        2
```

Такой IP-адрес уникален для узла в рамках всех доступных сетей (т. е. каждый узел, с которым вы можете связаться, должен иметь уникальный IP-адрес). Чтобы различать узлы, находящиеся внутри и извне сети, IP-адрес подразделяется на две части: часть *сети* и часть *узла*.

Это разделение записывается с помощью *маски сети* — набора единиц, за которым следует набор нулей. Часть IP-адреса, которая попадает на единицы — сетевая, оставшаяся часть — узловая. Как обычно,

маска сети может записываться в виде IP-адреса.

Листинг 3.9: Пример разделения сети/узла

```
IP-адрес:      192      168      0         2
            11000000 10101000 00000000 00000010
Маска сети: 11111111 11111111 11111111 00000000
               255      255      255       0
           +--------------------------+--------+
                      Сеть               Узел
```

Другими словами, 192.168.0.14 — все еще входит в состав сети из нашего примера, а 192.168.1.2 — уже нет.

Широковещательный адрес — это IP-адрес с такой же сетевой частью, как у вашей сети, но у которого узловая часть состоит только из единиц. Каждый узел вашей сети слушает этот IP-адрес. Он действительно предназначен для широковещательной рассылки пакетов.

Листинг 3.10: Широковещательный адрес

```
IP-адрес:             192      168      0         2
                   11000000 10101000 00000000 00000010
Широковещательный  11000000 10101000 00000000 11111111
адрес:                192      168      0        255
                  +--------------------------+--------+
                               Сеть              Узел
```

Чтобы бороздить просторы интернета, вы должны знать, через какой узел происходит подключение к интернету. Этот узел называется *шлюзом*. Так как это обычный узел, ему присвоен обычный IP-адрес (например, 192.168.0.1).

Выше мы говорили, что каждому узлу присваивается свой собственный IP-адрес. Чтобы связываться с узлом по имени (вместо IP-адреса), нужна служба, которая переводит имя (такое, как *dev.gentoo.org*) в IP-адрес (например, *64.5.62.82*). Такая служба называется службой имен. Чтобы пользоваться ей, нужно указать необходимые *серверы имен* в `/etc/resolv.conf`.

Иногда ваш шлюз сам является сервером имен. В остальных случаях вам необходимо указывать серверы имен, предоставляемые поставщиком интернета.

В итоге, для дальнейшего вам потребуется следующая информация:

Параметр настройки сети	Пример
Ваш IP-адрес	**192.168.0.2**
Маска сети	**255.255.255.0**
Широковещательный адрес	**192.168.0.255**
Шлюз	**192.168.0.1**
Сервер(ы) имен	**195.130.130.5, 195.130.130.133**

Использование ifconfig и route

Настройка вашей сети состоит из трех шагов. Сначала мы назначаем себе IP-адрес с помощью `ifconfig`. Затем мы настраиваем маршрутизацию к шлюзу, пользуясь `route`. И в завершение мы помещаем IP-адреса серверов имен в `/etc/resolv.conf`.

Для назначения IP-адреса потребуется ваш IP-адрес, широковещательный адрес и маска сети. Узнав их, выполните следующую команду, заменив `${IP_ADDR}` на свой IP-адрес, `${BROADCAST}` на свой широковещательный адрес, а `${NETMASK}` на свою маску сети:

Листинг 3.11: Использование ifconfig

```
# ifconfig eth0 ${IP_ADDR} broadcast ${BROADCAST} netmask ${NETMASK} up
```

Теперь настройте маршрутизацию с помощью `route`. Подставьте IP-адрес своего шлюза вместо `${GATEWAY}`:

Листинг 3.12: Использование route

```
# route add default gw ${GATEWAY}
```

Затем откройте `/etc/resolv.conf` в своем любимом редакторе (в нашем примере используется `nano`):

Листинг 3.13: Создание /etc/resolv.conf

```
# nano -w /etc/resolv.conf
```

Заполните данные своих серверов имен по следующему образцу. Обязательно замените `${NAMESERVER1}` и `${NAMESERVER2}` на соответствующие адреса серверов имен:

Листинг 3.14: Образец /etc/resolv.conf

```
nameserver ${NAMESERVER1}
nameserver ${NAMESERVER2}
```

Готово. Теперь проверьте свою сеть, «попинговав» какой-либо сервер интернета (например, Google). Если все заработало, примите наши поздравления! Теперь вы готовы к установке Gentoo. Приступайте к подготовке дисков.

4. Подготовка дисков

4.a. Общие сведения о блочных устройствах

Блочные устройства

Мы достаточно подробно рассмотрим аспекты работы с дисками в Gentoo Linux и Linux вообще, включая файловые системы, разделы и блочные устройства. Позже, когда вы уже освоитесь с дисками и файловыми системами, мы проведем вас через процесс настройки разделов и файловых систем для установки Gentoo Linux.

Для начала, давайте познакомимся с *блочными устройствами*. Вероятно, наиболее известно блочное устройство, соответствующее в системе Linux первому IDE-диску, а именно `/dev/hda`. А если в вашей системе используются диски SCSI или SATA, то первым жестким диском будет `/dev/sda`.

Подобные блочные устройства представляют собой абстрактный интерфейс к диску. Прикладные программы могут использовать их для обращения к дискам, не беспокоясь, к какому типу те принадлежат: IDE, SCSI или какому-то еще. Программы могут просто обращаться к накопителю, как к набору смежных 512-байтных блоков с произвольным доступом.

Разделы

Хотя теоретически для размещения системы Linux диск можно использовать целиком, так почти никогда не делают на практике. Вместо этого большое физическое блочное устройство разбивают на меньшие блочные устройства, более удобные для обращения. В архитектуре x86 они называются *разделами*.

Существуют разделы трех типов: *первичные* (primary), *расширенные* (extended) и *логические* (logical).

Первичный раздел — это раздел, информация о котором хранится в MBR (Master Boot Record — главной загрузочной записи). Так как MBR очень мала (512 байт), можно определять всего четыре первичных раздела (например, от `/dev/hda1` до `/dev/hda4`).

Расширенный раздел — это особый первичный раздел (имеется в виду, что расширенный раздел должен быть одним из четырех возможных первичных разделов), в котором содержатся другие разделы. Таких разделов изначально не существовало, но их введение помогло расширить существующую схему разметки без потери совместимости, как только четырех разделов перестало хватать.

Логический раздел — это раздел, входящий в расширенный раздел. Его определение находится не в MBR, а внутри расширенного раздела.

Дополнительные возможности

На установочном компакт-диске для x86 предусмотрена поддержка EVMS и LVM2. Использование EVMS и LVM2 повышает гибкость разбиения диска. В ходе установки мы останавливаемся на «обычных» разделах, но стоит запомнить, что EVMS и LVM2 тоже поддерживаются.

4.b. Разработка схемы разбиения диска

Схема разбиения по умолчанию

Если вам не интересно заниматься разработкой схемы для своей системы, можно воспользоваться схемой, используемой в этой книге:

Раздел	Файловая система	Размер	Описание
`/dev/hda1`	**ext2**	**32МБ**	**загрузочный раздел**
`/dev/hda2`	**(swap)**	**512МБ**	**раздел подкачки**
`/dev/hda3`	**ext3**	**оставшаяся часть диска**	**корневой раздел**

Если вам интересно узнать, какого размера должны быть разделы, и сколько их вам вообще может потребоваться, читайте дальше. В противном случае приступайте к созданию разделов, описанному в главе использование fdisk для создания разделов.

Сколько и каких именно?

Количество разделов очень сильно зависит от назначения системы. Например, если у вас много пользователей, вам, скорее всего, захочется отделить `/home` для повышения безопасности и упрощения резервного копирования. Если вы устанавливаете Gentoo в роли почтового сервера, следует отделить `/var`, т.к. вся почта хранится там. Затем, правильно выбрав файловую систему, вы добьетесь максимальной производительности. Для игровых серверов потребуется отдельный раздел `/opt`, так как большинство программ для их работы устанавливается туда. Причины выделения те же, что и для `/home`: безопасность и резервное копирование. Определенно не помешает побольше места для `/usr`: не только потому, что там хранится большинство приложений, а еще из-за того, что лишь дерево Portage, не считая размещенных в нем архивов с исходными кодами, занимает около 500 МБ.

Как видите, все зависит от ваших целей. Наличие отдельных разделов или томов имеет следующие плюсы:

- для каждого раздела или тома можно выбрать наиболее подходящую файловую систему
- свободное место во всей системе не кончится «вдруг» из-за того, что одна-единственная сбойная программа постоянно записывает файлы в раздел или том
- необходимая проверка файловых систем будет занимать меньше времени, т.к. проверка разных разделов может выполняться параллельно (еще больший выигрыш времени дает использование нескольких физических дисков)
- можно повысить безопасность системы, монтируя часть разделов в режиме read-only (только для чтения), nosuid (игнорируется бит setuid), noexec (игнорируется бит исполнения) и т.д.

Однако, у создания множества разделов есть один большой минус: при неправильной настройке можно получить систему, в которой много свободного места на одном разделе, и совершенно нет на другом. Кроме того, на дисках SCSI и SATA возможно создание не более 15 разделов.

Для примера мы покажем разбиение диска объемом 20ГБ, используемого в демонстрационном ноутбуке (с веб-сервером, почтовым сервером, средой Gnome и т.д.):

Листинг 2.1: Пример файловой системы

```
$ df -h
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/hda5     ext3    509M  132M  351M  28% /
```

```
/dev/hda2      ext3    5.0G  3.0G  1.8G  63% /home
/dev/hda7      ext3    7.9G  6.2G  1.3G  83% /usr
/dev/hda8      ext3   1011M  483M  477M  51% /opt
/dev/hda9      ext3    2.0G  607M  1.3G  32% /var
/dev/hda1      ext2     51M   17M   31M  36% /boot
/dev/hda6      swap    516M   12M  504M   2% <not mounted>
(свободное место для будущего использования: 2 ГБ)
```

Раздел `/usr`, как видим, почти заполнен (использовано 83%), но когда все программы установлены, `/usr` растет не слишком быстро. Хотя отведение нескольких гигабайт дискового пространства для `/var` может показаться расточительством, помните, что этот раздел по умолчанию используется Portage для компиляции пакетов. Если вы захотите удержать `/var` в рамках более разумного размера, например, 1ГБ, вам потребуется изменить переменную `PORTAGE_TMPDIR` в `/etc/make.conf`, чтобы она указывала на раздел, где достаточно свободного места для компиляции чрезвычайно больших пакетов, таких как OpenOffice.

4.c. Использование fdisk для создания разделов

В следующих разделах описывается, как разбить диск в соответствии с примерной схемой, описанной ранее, а именно:

Раздел	Описание
/dev/hda1	**загрузочный раздел**
/dev/hda2	**раздел подкачки**
/dev/hda3	**корневой раздел**

Эту схему вы можете изменять по своему усмотрению.

Просмотр текущей схемы разбиения диска

`fdisk` — это популярная и очень мощная утилита для создания разделов на ваших дисках. Запустите `fdisk`, указав свой диск в качестве параметра (в примере мы используем `/dev/hda`):

Листинг 3.1: Запуск fdisk

```
# fdisk /dev/hda
```

После запуска `fdisk` выдаст такое приглашение:

Листинг 3.2: Приглашение fdisk

```
Command (m for help):
```

Нажмите `p`, чтобы вывести текущую схему разбиения диска:

Листинг 3.3: Пример схемы разделов диска

```
Command (m for help): p

Disk /dev/hda: 240 heads, 63 sectors, 2184 cylinders
Units = cylinders of 15120 * 512 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1              1        14    105808+  83  Linux
/dev/hda2             15        49    264600   82  Linux swap
/dev/hda3             50        70    158760   83  Linux
/dev/hda4             71      2184  15981840    5  Extended
/dev/hda5             71       209   1050808+  83  Linux
/dev/hda6            210       348   1050808+  83  Linux
/dev/hda7            349       626   2101648+  83  Linux
/dev/hda8            627       904   2101648+  83  Linux
/dev/hda9            905      2184   9676768+  83  Linux
```

```
Command (m for help):
```

В данном случае на диске есть семь разделов Linux (для которых в графе «System» указано «Linux») и один раздел подкачки (в списке показан как «Linux swap»).

Удаление всех разделов

Сначала удалим с диска все существующие разделы. Для удаления раздела вводите `d`. Например для удаления существующего `/dev/hda1`:

Листинг 3.4: Удаление раздела

```
Command (m for help): d
Partition number (1-4): 1
```

Удаление раздела будет запланировано. Он больше не будет отображаться при вводе `p`, но фактически не будет удалятся до тех пор, пока вы не сохраните внесенные изменения. Если вы ошиблись и хотите прервать разбиение без сохранения изменений, немедленно введите `q` и нажмите ENTER; тогда ваш раздел сохранится в неизменном виде.

Теперь, если вы действительно собираетесь удалить в своей системе все разделы, повторяйте ввод `p` для показа оставшихся разделов, затем `d` и номера удаляемого раздела до тех пор, пока разделы не кончатся. В итоге вы получите пустую таблицу разделов:

Листинг 3.5: Пустая таблица разделов

```
Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System

Command (m for help):
```

Теперь, когда мы очистили таблицу разделов, хранящуюся в оперативной памяти, настало время создавать разделы. Мы воспользуемся примерной схемой разбиения, описанной выше. Естественно, не следуйте этим инструкциям буквально, если только вам не нужна таблица разделов, идентичная нашей!

Создание загрузочного раздела

Сначала создадим маленький загрузочный раздел. Введите `n` для создания нового раздела, затем `p`, чтобы выбрать первичный раздел, и `1` для указания первого первичного раздела. На вопрос о первом цилиндре нажмите ввод. На вопрос о последнем цилиндре введите `+32M`, чтобы создать раздел размером 32МБ:

Листинг 3.6: Создание загрузочного раздела

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-3876, default 1): (нажмите ввод)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-3876, default 3876): +32M
```

Теперь, введя `p`, вы должны увидеть следующий текст:

Листинг 3.7: Созданный загрузочный раздел

```
Command (m for help): p
```

```
Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1          1        14    105808+  83  Linux
```

Нам необходимо сделать этот раздел загружаемым. Введите `a` для включения признака загрузки с раздела, затем нажмите `1`. Если снова ввести `p`, можно увидеть, что в столбце «boot» появился знак `*`.

Создание раздела подкачки

Теперь создадим раздел подкачки. Для этого введите `n` (создание нового раздела), затем `p` для выбора первичного раздела. Потом нажмите `2`, чтобы создать второй первичный раздел, в нашем случае `/dev/hda2`. На вопрос о первом цилиндре просто нажмите ввод. На вопрос о последнем — ответьте `+512M`, чтобы создать раздел размером 512МБ. Сделав это, введите `t` для указания типа раздела, `2`, для выбора только что созданного раздела, и `82`, чтобы установить тип раздела «Linux Swap». При нажатии `p` по завершении этих действий, таблица разделов должна выглядеть примерно так:

Листинг 3.8: Список разделов после создания раздела подкачки

```
Command (m for help): p

Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1 *        1        14    105808+  83  Linux
/dev/hda2         15        81    506520   82  Linux swap
```

Создание корневого раздела

Теперь создадим корневой раздел. Для этого введите `n` (создание нового раздела), затем `p` (первичный раздел). После этого нажмите `3` для создания третьего первичного раздела, в нашем случае `/dev/hda3`. На вопрос о первом цилиндре нажмите ввод. На вопрос о последнем — также нажмите ввод, чтобы раздел занял все оставшееся свободное место на диске. По завершении этих шагов, при вводе `p` должна выводиться подобная таблица разделов:

Листинг 3.9: Список разделов после создания корневого раздела

```
Command (m for help): p

Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1 *        1        14    105808+  83  Linux
/dev/hda2         15        81    506520   82  Linux swap
/dev/hda3         82      3876  28690200   83  Linux
```

Сохранение схемы разбиения

Для сохранения схемы разбиения и выхода из `fdisk`, введите `w`.

Листинг 3.10: Сохранение и выход из fdisk

```
Command (m for help): w
```

Теперь, создав все разделы, перейдем к созданию файловых систем.

4.d. Создание файловых систем

Введение

Разделы созданы, настало время разместить на них файловые системы. Если вам безразлично, какую файловую систему использовать, и вы вполне довольны той, что мы предлагаем в книге по умолчанию, переходите к размещению файловой системы в разделе. Если нет, читайте дальше, чтобы разузнать о существующих файловых системах...

Файловые системы?

В ядре Linux поддерживаются различные файловые системы. Мы расскажем о самых распространенных из них: ext2, ext3, ReiserFS, XFS и JFS.

ext2 — испытанная файловая система Linux, в которой отсутствуют средства журналирования метаданных. Поэтому ее рядовая проверка при запуске может быть очень долгой. Сегодня существует довольно много журналируемых файловых систем нового поколения, целостность которых проверяется очень быстро, и поэтому обычно есть смысл использовать их. Журналируемые файловые системы позволяют избежать долгих задержек при запуске, когда состояние файловой системы неустойчиво.

ext3 — это журналируемая версия ext2, где для ускорения восстановления появилось журналирование метаданных, а также другие режимы, наподобие полного или упорядоченного журналирования данных. ext3 — очень хорошая и надежная файловая система. В ней есть дополнительная возможность индексации через хэшированные сбалансированные деревья (b-tree), что в большинстве случаев обеспечивает высокую скорость. Такую индексацию можно включить, добавив к команде `mke2fs` параметр `-O dir_index`. Короче говоря, ext3 — превосходная файловая система.

ReiserFS — файловая система, построенная на B*-деревьях. У нее очень хорошая скорость, и она намного (часто в 10-15 раз) быстрее ext2 и ext3 при работе с маленькими файлами (менее 4КБ). ReiserFS также великолепно масштабируется; в ней есть средства журналирования. На дату выхода ядра 2.4.18, ReiserFS признана стабильной и пригодной как для систем общего назначения, так и для крайних случаев типа создания больших томов, использования множества маленьких файлов, гигантских файлов или же каталогов с десятками тысяч файлов.

XFS — это файловая система с журналированием метаданных и дополнительными средствами, оптимизированными для работы с максимальным объемом данных. Ее применение рекомендуется только в Linux-системах с высококлассными дисками SCSI или дисками с оптическим подключением (fibre-channel), при наличии источников бесперебойного питания. Из-за того, что XFS выполняет очень агрессивное промежуточное кэширование в оперативной памяти, при внезапном отключении питания очень вероятна потеря небрежно спроектированными программами (не принимающими адекватные меры предосторожности при записи файлов на диск, и таких довольно много) изрядного количества данных.

JFS — высокопроизводительная журналируемая файловая система производства IBM. Она лишь недавно стала пригодной для широкого применения, и пока слишком мало данных, чтобы сказать что-то хорошее или плохое о ее общей стабильности.

Размещение файловой системы в разделе

Для создания в разделе или томе файловой системы каждого типа существуют специальные средства:

Файловая система	Команда создания
ext2	**`mke2fs`**
ext3	**`mke2fs -j`**
reiserfs	**`mkreiserfs`**
xfs	**`mkfs.xfs`**
jfs	**`mkfs.jfs`**

Например, чтобы у загрузочного раздела (`/dev/hda1` в наших примерах) была файловая система ext2, а у корневого раздела (`/dev/hda3` в наших примерах) — ext3, требуется выполнить:

Листинг 4.1: Создание файловых систем разделов

```
# mke2fs /dev/hda1
# mke2fs -j /dev/hda3
```

Теперь самостоятельно создайте файловые системы на своих только что созданных разделах (логических томах).

Подключение раздела подкачки

`mkswap` — команда, используемая для инициализации разделов подкачки:

Листинг 4.2: Создание идентификатора раздела подкачки

```
# mkswap /dev/hda2
```

Для подключения раздела подкачки воспользуйтесь `swapon`:

Листинг 4.3: Подключение раздела подкачки

```
# swapon /dev/hda2
```

Теперь командами, приведенными выше, создайте и подключите раздел подкачки в своей системе.

4.e. Монтирование

Теперь, когда разделы созданы, а файловые системы размещены, настало время смонтировать (подключить к системе) эти разделы. Используйте команду `mount`. Не забудьте создать для каждого раздела соответствующие каталоги монтирования. Например, смонтируем корневой и загрузочный разделы:

Листинг 5.1: Монтирование разделов

```
# mount /dev/hda3 /mnt/gentoo
# mkdir /mnt/gentoo/boot
# mount /dev/hda1 /mnt/gentoo/boot
```

Примечание: Если вы хотите разместить каталог `/tmp` в отдельном разделе, не забудьте изменить права доступа к этому каталогу после монтирования: `chmod 1777 /mnt/gentoo/tmp`. Это также относится к `/var/tmp`.

Еще нам потребуется смонтировать файловую систему proc (виртуальный интерфейс ядра) в каталог `/proc`. Но сначала надо поместить в разделы нужные файлы.

Переходите к установке установочных файлов Gentoo.

5. Установка установочных файлов Gentoo

5.a. Установка архива стадии

Установка текущей даты и времени

Перед тем, как продолжать, убедитесь в правильности системной даты и времени, и при необходимости обновите их. Если часы установлены неверно, это в будущем может привести к странным результатам!

Для проверки времени, введите команду `date`:

Листинг 1.1: Проверка даты и времени

```
# date
Fri Mar 29 16:21:18 UTC 2005
```

Если часы установлены неправильно, измените время командой `date ММДДччммГГГГ` (формат: **М**есяц, **Д**ень, **ч**асы, **м**инуты и **Г**од). На этом этапе следует указывать время UTC. Позднее вы сможете установить свой часовой пояс. Например, для установки часов на 29 марта, 16:21, 2005 год, выполните:

Листинг 1.2: Установка даты и времени

```
# date 032916212005
```

Выбор источника

Следующий необходимый шаг — установка архива *stage3* на вашу систему. Нужный архив можно скачать из интернета, или, если вы загрузились с универсального установочного диска Gentoo, скопировать с самого диска. Если у вас есть универсальный диск, а на нем — нужная стадия, то загрузка из интернета — лишняя трата трафика и времени, т.к. файлы стадии окажутся совершенно одинаковы. В большинстве случаев, определиться с выбором архива стадии вам поможет команда `uname -m`.

- по умолчанию: использование файла стадии из интернета
- альтернатива: использование файла стадии с установочного диска

5.b. По умолчанию: использование файла стадии из интернета

Загрузка архива стадии

Перейдите в точку подключения файловой системы Gentoo (обычно — `/mnt/gentoo`):

Листинг 2.1: Переход к точке подключения Gentoo

```
# cd /mnt/gentoo
```

Для загрузки файла стадии у вас уже есть различные инструменты, зависящие от типа диска, с которого производится установка. Если имеется `links`, можете сразу открыть список зеркал Gentoo и выбрать ближайшее.

Когда `links` нет, в вашем распоряжении должен оказаться `lynx`. Если требуется использовать прокси-сервер, экспортируйте переменные `http_proxy` и `ftp_proxy`:

Листинг 2.2: Настройка информации о прокси-сервере для lynx

```
# export http_proxy="http://proxy.server.com:port"
# export ftp_proxy="http://proxy.server.com:port"
```

Предположим, у вас под рукой есть `links`.

Перейдите в каталог `releases/`, в нем откройте каталог, соответствующий вашей архитектуре (например, `x86/`), затем каталог с версией Gentoo (`2006.1/`), и, наконец, каталог `stages/`. Здесь находятся все доступные архивы стадий для вашей архитектуры (они могут располагаться в подкаталогах с названиями разновидностей архитектуры). Выберите один из них и нажмите `D` для загрузки. По окончании — нажмите `Q`, чтобы выйти из браузера.

Листинг 2.3: Открытие списка зеркал в links

```
# links http://www.gentoo.org/main/ru/mirrors.xml

(если в links нужна поддержка прокси-сервера:)
# links -http-proxy proxy.server.com:8080 http://www.gentoo.org/main/ru/mirrors.xml
```

Удостоверьтесь, что вы скачали **stage3**-архив — установка с использованием файлов stage1 и stage2 более не поддерживается.

Если вы хотите проверить целостность загруженного архива стадии, с помощью `md5sum` сравните результат с контрольной суммой MD5, взятой с зеркала. Например, для проверки целостности архива стадии для x86:

Листинг 2.4: Пример проверки целостности архива стадии

```
# md5sum -c stage3-x86-2006.1.tar.bz2.DIGESTS
stage3-x86-2006.1.tar.bz2: OK
```

Распаковка архива стадии

Распакуйте загруженный архив стадии в своей системе. Мы используем `tar`, т.к. это простейший способ:

Листинг 2.5: Распаковка архива стадии

```
# tar xvjpf stage3-*.tar.bz2
```

Убедитесь, что вы используете точно такие же параметры командной строки (`xvjpf`). Значения параметров: `x` — *извлечение*, `v` — *подробные сообщения*, чтобы видеть, что происходит во время распаковки (необязательный параметр), `j` — *декомпрессия bzip2*, `p` — *сохранение прав доступа*, и, наконец, `f` — указывает на то, что мы распаковываем файл, а не то, что подается на стандартный ввод.

Примечание: На установочных компакт-дисках и загрузочных образах для некоторых архитектур (например, MIPS) `tar` встроен в оболочку BusyBox, которая на данный момент не поддерживает параметр `v`. Тогда вместо приведенных указывайте параметры `xjpf`.

После развертывания установки стадии переходите к установке дерева портежей.

5.c. Альтернатива: использование файла стадии с установочного диска

Извлечение архива стадии

Важно: Если вы — в x86 и используете «живой диск» с установщиком, на нем *нет* стадий. Вам понадобится последовать указаниям по использованию файла стадии из интернета.

Архивы стадий находятся на компакт-диске в каталоге `/mnt/cdrom/stages`. Для получения списка доступных стадий, воспользуйтесь `ls`:

Листинг 3.1: Список доступных архивов стадий

```
# ls /mnt/cdrom/stages
```

Если система сообщает об ошибке, возможно, сначала нужно смонтировать CD-ROM:

Листинг 3.2: Монтирование CD-ROM

```
# ls /mnt/cdrom/stages
ls: /mnt/cdrom/stages: No such file or directory
# mount /dev/cdroms/cdrom0 /mnt/cdrom
# ls /mnt/cdrom/stages
```

Перейдите в точку подключения Gentoo (обычно — `/mnt/gentoo`):

Листинг 3.3: Смена каталога на /mnt/gentoo

```
# cd /mnt/gentoo
```

Теперь утилитой `tar` распакуем выбранный архив стадии. Убедитесь, что вы используете точно такие же параметры (`xvjpf`)! Параметр `v` необязателен и может не работать в некоторых версиях `tar`. В следующем примере мы распакуем архив стадии `stage3-<архитектура>-2006.1.tar.bz2`. В качестве имени архива обязательно указывайте название файла нужной вам стадии.

Листинг 3.4: Распаковка архива стадии

```
# tar xvjpf /mnt/cdrom/stages/stage3-<архитектура>-2006.1.tar.bz2
```

После установки стадии переходите к установке дерева портежей.

5.d. Установка дерева портежей

Распаковка снимка портежей

Теперь вам нужно установить снимок дерева портежей — набор файлов, содержащих сведения для Portage о программном обеспечении, доступном для установки, имеющихся профилях и т.д.

Загрузка и установка снимка дерева портежей

Перейдите к точке подключения вашей системы (обычно — `/mnt/gentoo`):

Листинг 4.1: Переход к точке подключения

```
# cd /mnt/gentoo
```

Запустите `links` (или `lynx`) и откройте списокк зеркал Gentoo. Выберите ближайшее зеркало и перейдите в каталог `snapshots/`. Оттуда загрузите новейший снимок дерева портежей (`portage-latest.tar.bz2`), выбрав его из списка и нажав `D`.

Листинг 4.2: Просмотр списка зеркал Gentoo

```
# links http://www.gentoo.org/main/ru/mirrors.xml
```

Закройте браузер нажатием `Q`. Теперь у вас в `/mnt/gentoo` появился снимок дерева портежей.

Если нужно убедиться в целостности загруженного снимка, воспользуйтесь `md5sum`, и сравните результат с контрольной суммой MD5, имеющейся на зеркале.

Листинг 4.3: Checking integrity of a Portage snapshot

```
# md5sum -c portage-latest.tar.bz2.md5sum
portage-latest.tar.bz2: OK
```

Теперь нужно распаковать снимок. Убедитесь в том, что вы используете в точности такие же параметры; последний параметр — заглавная `C`, а не строчная `c`.

Листинг 4.4: Извлечение снимка дерева портежей

```
# tar xvjf /mnt/gentoo/portage-latest.tar.bz2 -C /mnt/gentoo/usr
```

5.e. Настройка параметров компиляции

Введение

Для оптимизации Gentoo вы можете создать ряд переменных, которые повлияют на поведение Portage. Все их можно создавать как переменные среды (с помощью `export`), но тогда они будут лишь временными. Для хранения ваших настроек предназначен конфигурационный файл Portage, `/etc/make.conf`. Именно его мы сейчас будем редактировать.

Примечание: Список и описание всех допустимых переменных находятся в файле `/mnt/gentoo/etc/make.conf.example`. Для успешной установки Gentoo, достаточно установить значения только тех переменных, которые описаны ниже.

Запустите ваш любимый редактор (в этом руководстве используется `nano`) для изменения параметров

оптимизации, которые обсуждаются далее.

Листинг 5.1: Открытие /etc/make.conf

```
# nano -w /mnt/gentoo/etc/make.conf
```

Как вы могли заметить, у файла `make.conf.example` обычная структура: строки комментария начинаются со знака «#», а в остальных строках выражениями вида `ПЕРЕМЕННАЯ="значение"` определяются переменные. У файла `make.conf` такой же формат. Некоторые из переменных мы обсудим ниже.

CHOST

В переменной `CHOST` определяется, для какой архитектуры скомпилирована система. В ней уже должно быть установлено правильное значение. **Не изменяйте эту переменную**, т.к. это может повредить вашу систему. Если переменная `CHOST`, по-вашему, выглядит неправильно, возможно, вы используете не тот архив stage3.

CFLAGS и CXXFLAGS

Переменные `CFLAGS` и `CXXFLAGS` определяют параметры оптимизации компилятора `gcc` для языков C и C++, соответственно. При том, что общие значения следует устанавливать здесь, максимальная производительность достигается в том случае, когда для каждой компилируемой программы устанавливаются свои собственные оптимальные параметры. Причина в том, что все программы различны.

В файле `make.conf` следует указывать параметры оптимизации, которые, на ваш взгляд, повысят скорость системы *в целом*. Не помещайте сюда экспериментальные значения; излишняя оптимизация может привести к плохому поведению программ (аварийным завершениям, или, что хуже, неправильной работе).

Мы не будем описывать все возможные параметры оптимизации. Если вы хотите узнать о них всё, почитайте онлайновые руководства GNU (англ.) или страницу описания `gcc` (`info gcc` — доступна только на работающей системе Linux). Не забудьте прочитать сам файл `make.conf.example`: в нем также есть немало примеров и полезных сведений.

Первый параметр — флаг `-march=`, где указывается название целевой архитектуры. Возможные варианты описаны в комментариях в файле `make.conf.example`. Например, для архитектуры x86 Athlon XP:

Листинг 5.2: Значение -march для GCC

```
# пользователям AMD64, желающим получить действительно 64-битную
# систему, следует использовать -march=k8
# пользователи EM64T должны указать -march=nocona
-march=athlon-xp
```

Следующий — флаг `-O` (заглавная латинская «O», а не ноль), определяющий класс оптимизации в `gcc`. Допустимые значения: `s` — оптимизация по размеру; `0` — ноль, без оптимизации; `1`, `2` или `3` — все большая оптимизация по скорости (в каждый класс входят все флаги предыдущего, и некоторые дополнительные). Например, для оптимизации класса 2:

Листинг 5.3: Значение -O для GCC

```
-O2
```

Другой популярный флаг оптимизации — `-pipe` (использование для связи между различными проходами компилятора каналов вместо временных файлов).

Заметьте, что использование `-fomit-frame-pointer` (не хранить в регистре указатель стекового кадра для функций, которым он не нужен) может всерьез помешать отладке программ!

При определении переменных `CFLAGS` и `CXXFLAGS`, нужно объединять несколько флагов оптимизации, как в следующем примере:

Листинг 5.4: Определение переменных CFLAGS и CXXFLAGS

```
CFLAGS="-march=athlon-xp -pipe -O2"  # для пользователей AMD64: -march=k8
                                     # для пользователей EM64T: -march=nocona
CXXFLAGS="${CFLAGS}"                 # указывайте одинаковые значения
                                     # обеих переменных
```

MAKEOPTS

С помощью `MAKEOPTS` определяется, сколько параллельных процессов компиляции можно запускать при установке пакета. Хороший (но не всегда идеальный) вариант — значение, равное количеству процессоров в системе плюс один.

Листинг 5.5: MAKEOPTS для обычной однопроцессорной системы

```
MAKEOPTS="-j2"
```

На старт, внимание, марш!

Измените `/mnt/gentoo/etc/make.conf` в соответствии со своими пожеланиями, и сохраните изменения (пользователям `nano` нужно нажать `CTRL+X`). Теперь вы готовы к установке базовой системы Gentoo.

6. Установка базовой системы Gentoo

6.a. Изменение корневого каталога

Дополнительно: выбор зеркала

Для ускорения загрузки исходного кода рекомендуется выбрать быстрое зеркало. Portage ищет переменную GENTOO_MIRRORS в файле `make.conf` и использует зеркала, перечисленные в ней. Конечно, можно просмотреть наш список зеркал и выбрать одно или несколько, географически ближайших к вам (чаще всего они и будут самыми быстрыми), но мы предлагаем вам удобную утилиту `mirrorselect`, которая позволяет выбрать желаемые зеркала более удобным способом.

Листинг 1.1: Запуск mirrorselect для установки переменной GENTOO_MIRRORS

```
# mirrorselect -i -o >>/mnt/gentoo/etc/make.conf
```

Предупреждение: Не выбирайте зекрала IPv6. Файлы стадий пока не поддерживают протокол IPv6.

Вторая важная настройка — установка значения переменной SYNC в файле `make.conf`. Эта переменная указывает на сервер rsync (сервер удаленной синхронизации), выбранный вами для обновления дерева Portage (коллекции файлов ebuild — сборочных сценариев, содержащих все данные, нужные Portage для скачивания и установки программ). Хотя вписать адрес сервера в SYNC можно и вручную, `mirrorselect` позволяет упростить это действие:

Листинг 1.2: Выбор зеркала rsync с помощью mirrorselect

```
# mirrorselect -i -r -o >> /mnt/gentoo/etc/make.conf
```

После выполнения `mirrorselect`, мы рекомендуем проверить все значения в файле `/mnt/gentoo/etc/make.conf`!

Копирование сведений о DNS

Перед тем, как перейти в новую среду, осталось сделать одно дело: скопировать информацию о DNS (системе доменных имен) в файл `/etc/resolv.conf`. Это нужно, чтобы при переходе сохранить работоспособность сети. В файле `/etc/resolv.conf` содержатся адреса серверов имен, используемых в вашей сети.

Листинг 1.3: Копирование информации о DNS

```
(параметр «-L» нужен, чтобы случайно не скопировать
 символьную ссылку)
# cp -L /etc/resolv.conf /mnt/gentoo/etc/resolv.conf
```

Монтирование файловых систем /proc и /dev

Смонтируйте файловую систему `/proc` в `/mnt/gentoo/proc`, чтобы после изменения корневого каталога устанавливаемая система смогла обращаться к информации, предоставляемой ядром, а затем создайте привязку монтирования файловой системы `/dev`.

Листинг 1.4: Монтирование /proc и /dev

```
# mount -t proc none /mnt/gentoo/proc
# mount -o bind /dev /mnt/gentoo/dev
```

Переход в новую среду

Итак, все разделы подготовлены, а базовая операционная среда установлена. Теперь пора войти в нее, *изменив корневой каталог*. Таким образом, мы перейдем из текущей установочной среды (с компакт-диска или другого установочного носителя) в свою устанавливаемую систему (находящуюся в недавно размеченных разделах).

Изменение выполняется в три этапа. Сначала мы с помощью `chroot` изменим корневой каталог с `/` (находящийся на установочном носителе) на `/mnt/gentoo` (находящийся на ваших дисковых разделах). Затем мы создадим новую среду, пользуясь утилитой `env-update`, которая, собственно, создает переменные среды. Наконец, мы загрузим эти переменные в память при помощи `source`.

Листинг 1.5: Изменение корневого каталога для перехода в новую среду

```
# chroot /mnt/gentoo /bin/bash
# env-update
>> Regenerating /etc/ld.so.cache...
# source /etc/profile
# export PS1="(chroot) $PS1"
```

Поздравляем! Теперь вы в своей собственной среде Gentoo Linux. Конечно, на этом она далеко еще не готова, поэтому в руководстве еще осталось несколько разделов :-)

6.b. Настройка Portage

Обновление дерева Portage

Теперь надо обновить дерево Portage до самой последней версии с помощью команды `emerge --sync`:

Листинг 2.1: Обновление дерева портежей

```
# emerge --sync
(если вы пользуетесь медленным терминалом, например, последовательным
 терминалом или кадровым буфером, для ускорения процесса можно добавить
 параметр --quiet :)
# emerge --sync --quiet
```

Если компьютер подключен к интернету через межсетевой экран, блокирующий прохождение rsync-пакетов, вы можете воспользоваться командой `emerge-webrsync`, которая скачивает и устанавливает снимок дерева.

Если выдано предупреждение, что имеется новая версия Portage и ее нужно обновить, выполните обновление командой `emerge portage`.

Выбор нужного профиля

Сначала дадим небольшое определение.

Профиль — конструктивный элемент любой системы Gentoo. В нем указываются не только значения по умолчанию для CHOST, CFLAGS и других важных переменных, он также привязывает систему к определенному диапазону допустимых версий пакетов. Диапазоны поддерживаются разработчиками Gentoo.

Раньше пользователь редко касался профиля. Однако, пользователи x86, hppa и alpha могут выбирать из двух вариантов профиля: одного для ядра 2.4, другого для ядра 2.6. Это вызвано необходимостью улучшения интеграции ядер 2.6. Для архитектур ppc и ppc64 также существует несколько профилей. Мы поговорим о них позже.

Узнать, какой профиль используется в системе в данный момент, вы можете командой:

Листинг 2.2: Выяснение используемого профиля

```
# ls -FGg /etc/make.profile
lrwxrwxrwx  1 48 Apr  8 18:51 /etc/make.profile -> ../usr/portage/profiles/default-linux/x86/
```

Если вы используете одну из трех упомянутых архитектур, профиль по умолчанию даст вам систему с ядром 2.6. Это рекомендуется по умолчанию, но за вами сохраняется и право выбора другого профиля.

Для некоторых архитектур также предусмотрены подпрофили `desktop` и `server`. Загляните в профиль `2006.1/`, чтобы узнать, есть ли подпрофили для вашей архитектуры. Возможно, вы захотите заглянуть в `make.defaults` профиля `desktop`, чтобы определить, подходит ли он вам.

Некоторым пользователям, возможно, захочется установить систему, основанную на старом профиле, с ядром 2.4. Если для этого есть веские основания, сначала надо проверить, существует ли такой дополнительный профиль. На системах x86 это можно сделать следующей командой:

Листинг 2.3: Выяснение наличия дополнительного профиля

```
# ls -d /usr/portage/profiles/default-linux/x86/no-nptl/2.4
/usr/portage/profiles/default-linux/x86/no-nptl/2.4
```

В приведенном примере дополнительный профиль 2.4 существует (т.е. нет сообщений об отсутствующем файле или каталоге). Рекомендуется использовать профиль по умолчанию, но если вы все же хотите сменить его, это можно сделать так:

Листинг 2.4: Переключение профиля на 2.4

```
(убедитесь, что используете нужную архитектуру; этот пример - для x86)
# ln -snf /usr/portage/profiles/default-linux/x86/no-nptl/2.4 /etc/make.profile
(вывод списка файлов в профиле 2.4)
# ls -FGg /etc/make.profile/
total 12
-rw-r--r--  1 939 Dec 10 14:06 packages
-rw-r--r--  1 347 Dec  3  2004 parent
-rw-r--r--  1 573 Dec  3  2004 virtuals
```

Для архитектуры ppc в выпуске 2006.1 появилось несколько новых профилей:

Листинг 2.5: Профили для PPC

```
(базовый профиль PPC, подходит для всех PPC-машин, минимальный)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc32/2006.1 /etc/make.profile
(профиль для G3)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc32/2006.1/G3 /etc/make.profile
(профиль для G3 Pegasos)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc32/2006.1/G3/Pegasos/ /etc/make.profile
(профиль для G4 (Altivec))
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc32/2006.1/G4 /etc/make.profile
(профиль для G4 (Altivec) Pegasos)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc32/2006.1/G4/Pegasos/ /etc/make.profile
```

Для архитектуры ppc64 в выпуске 2006.1 также появилось несколько новых профилей:

Листинг 2.6: Профили для PPC64

```
(базовый профиль PPC64 с 64-битным режимом пользователя, для всех PPC64-машин)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/64bit-userland /etc/make.profi
(базовый профиль PPC64 с 32-битным режимом пользователя, для всех PPC64-машин)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/32bit-userland /etc/make.profi
(для каждого режима пользователя существуют подпрофили, как показано ниже.)
(«userland» необходимо заменять на нужный вариант режима из примеров выше)
(профиль 970 для JS20)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/(userland)/970 /etc/make.profi
(профиль для G5)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/(userland)/970/pmac /etc/make.
(профиль для POWER3)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/(userland)/power3 /etc/make.pr
(профиль для POWER4)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/(userland)/power4 /etc/make.pr
(профиль для POWER5)
# ln -snf /usr/portage/profiles/default-linux/ppc/ppc64/2006.1/(userland)/power5 /etc/make.pr
(многорежимные профили на дату выхода этого выпуска нестабильны)
```

Настройка переменной USE

`USE` («использовать») — одна из самых мощных переменных, имеющихся в распоряжении пользователей Gentoo. Она позволяет при компиляции программ включать или отключать поддержку определенных необязательных функций. Например, некоторые программы можно компилировать с поддержкой gtk или qt на выбор. Другие можно собирать, включая или отключая поддержку SSL. Некоторые программы можно даже компилировать с поддержкой кадрового буфера (svgalib) вместо поддержки X11 (X-сервера).

В большинстве дистрибутивов пакеты собраны с поддержкой практически всех мыслимых вариантов. Это увеличивает размер программ и время запуска, не говоря уже о чрезмерных зависимостях. В Gentoo вы сами можете определять, с какими возможностями следует компилировать пакет. Здесь играет роль переменная `USE`.

В переменной `USE` указываются ключевые слова, которые отражаются на параметрах компиляции. Например, параметр *ssl* включает компиляцию с поддержкой ssl всех программ, которые способны его поддерживать. *-X* отключает поддержку X-сервера (обратите внимание на предшествующий знак «минус»). Параметры *gnome gtk -kde -qt* обеспечивают компиляцию ваших программ с поддержкой Gnome и gtk, но без поддержки KDE и qt, делая систему оптимальной для GNOME.

Настройки `USE` по умолчанию хранятся в файлах `make.defaults` вашего профиля. Файлы `make.defaults` находятся в каталоге, на который указывает `/etc/make.profile`, а также во всех родительских каталогах. Значение `USE` по умолчанию — это сумма всех значений `USE` во всех файлах `make.defaults`. Все, что вы вносите в `/etc/make.conf`, рассчитывается относительно этих значений. Когда вы добавляете что-либо к значению `USE`, оно добавляется в список по умолчанию. Когда удаляете что-либо (указывая ключевое слово с предшествующим знаком минус), оно удаляется из списка по умолчанию (если оно там вообще было). *Никогда* ничего не меняйте в `/etc/make.profile`. Все, что там находится, перезаписывается при обновлении Portage!

Полное описание переменной `USE` находится во второй части настольной книги Gentoo в главе USE-флаги. Полное описание возможных значений признаков использования находится в вашей системе в файле `/usr/portage/profiles/use.desc`.

Листинг 2.7: Просмотр допустимых USE-флагов

```
# less /usr/portage/profiles/use.desc
(для прокрутки пользуйтесь клавишами стрелок, для выхода нажимайте "q")
```

Например, приведем значение `USE` для системы, базирующейся на KDE, с включением поддержки DVD, ALSA и записи CD:

Листинг 2.8: Открытие файла /etc/make.conf

```
# nano -w /etc/make.conf
```

Листинг 2.9: Значение USE

```
USE="-gtk -gnome qt kde dvd alsa cdr"
```

Дополнительно: региональные параметры GLIBC

Скорее всего, вы будете использовать в системе один-два набора региональных параметров. Нужные регионы можно указать в `/etc/locale.gen`.

Листинг 2.10: Открытие /etc/locale.gen

```
# nano -w /etc/locale.gen
```

Вот пример одновременного подключения как английского (США), так и немецкого (Германия) с соответствующими кодировками (например, UTF-8).

Листинг 2.11: Укажите свои региональные настройки

```
en_US/ISO-8859-1
en_US.UTF-8/UTF-8
de_DE/ISO-8859-1
de_DE@euro/ISO-8859-15
```

Следующий шаг — запустить `locale-gen`. Это сгенерирует настройки для всех регионов, указанных вами в файле `/etc/locale.gen`.

Примечание: `locale-gen` имеется в `glibc-2.3.6-r4` и более новой. Если у вас старшая версия glibc, сейчас ее следует обновить.

Примечание: Настройки, подобной приведенной, для поддержки русского языка недостаточно. За дополнительными сведениями обращайтесь к отдельным руководствам по русификации Gentoo. — *прим. пер.*

Теперь приступим к настройке ядра.

7. Настройка ядра

7.a. Часовой пояс

Сначала необходимо выбрать часовой пояс (time zone), чтобы система знала, где вы находитесь. Найдите подходящий пояс в `/usr/share/zoneinfo`, затем скопируйте его в `/etc/localtime`. Постарайтесь, пожалуйста, избегать использования часовых поясов `/usr/share/zoneinfo/Etc/GMT*`, т. к. их названия не соответствуют ожидаемым поясам. Например, `GMT-8` фактически является поясом GMT+8.

Листинг 1.1: Установка часового пояса

```
# ls /usr/share/zoneinfo
(допустим, вы хотите использовать GMT)
# cp /usr/share/zoneinfo/GMT /etc/localtime
```

7.b. Установка исходных кодов ядра

Выбор ядра

Ядро Linux — это то, вокруг чего формируются все дистрибутивы. Это слой между пользовательскими программами и вашим оборудованием. Пользователи Gentoo могут выбирать из нескольких типов ядра. Их полный список и описание находится в руководстве по выбору ядра Gentoo (англ.).

Для x86 систем, помимо прочего, есть такие ядра, как `vanilla-sources` (ядро по умолчанию от разработчиков linux ядра), `gentoo-sources` (ядро с заплатками, улучшающими производительность).

С помощью команды `emerge` выберите и установите ядро. `USE="-doc"` требуется, чтобы на этом этапе избежать установки xorg-x11 или других связанных пакетов. `USE="symlink"` при новой установке не требуется, но обеспечивает правильное создание символьной ссылки `/usr/src/linux`:

Листинг 2.1: Установка исходных кодов ядра

```
# USE="-doc symlink" emerge gentoo-sources
```

Просмотрев содержимое каталога `/usr/src`, вы увидите символьную ссылку `linux`, которая указывает на каталог с исходными кодами ядра. В данном случае, установленный исходный код ядра указывает на `gentoo-sources-2.6.12-r10`. Имейте в виду, что ваша версия может отличаться.

Листинг 2.2: Просмотр папки с исходными кодами ядра

```
# ls -l /usr/src/linux
lrwxrwxrwx    1 root     root           12 Oct 13 11:04 /usr/src/linux -> linux-2.6.12-gentoo
```

Теперь настало время настройки и компиляции ядра. Можно использовать сценарий `genkernel`, который сформирует стандартное ядро, аналогичное используемому на установочном компакт-диске. Однако сначала мы расскажем о «ручной» настройке ядра, так как это лучший из способов оптимизации системы.

Если вы собираетесь настроить ядро вручную, то переходите к разделу по умолчанию: ручная настройка. Если вы хотите использовать `genkernel`, то читайте альтернатива: использование genkernel.

7.c. По умолчанию: ручная настройка

Введение

Ручная настройка ядра считается одним из самых сложных вопросов для пользователей Linux. Зато после настройки нескольких вариантов ядра вы и не вспомните, что это когда-то было трудно ;)

Однако есть одна непреложная *истина*: запуская ручную настройку ядра, следует знать состав аппаратуры своего компьютера. Большую часть информации можно собрать, установив пакет pciutils (`emerge pciutils`), в который входит `lspci`. `lspci` можно использовать, находясь в среде с временным корневым каталогом. Можно не обращать внимания на предупреждения, касающиеся *pcilib* (например: pcilib: cannot open /sys/bus/pci/devices [pcilib: не могу открыть /sys/bus/pci/devices]), которые выводит на экран `lspci`. Можно запустить `lspci` и *вне временной среды установки*. Результат тот же. Вы также можете запустить `lsmod` для просмотра модулей ядра, используемых установочным компакт-диском (это даст хорошую подсказку, какие параметры включать).

Теперь войдите в каталог с исходными кодами ядра и введите команду `make menuconfig`. Этой командой вы вызовете меню настройки, использующее ncurses.

Листинг 3.1: Вызов меню настройки

```
# cd /usr/src/linux
# make menuconfig
```

Перед вами появятся несколько разделов настройки. Сначала перечислим пункты, включение которых обязательно (иначе Gentoo не заработает, или для его работы потребуются дополнительные ухищрения).

Включение нужных пунктов

Прежде всего, разрешите использование разрабатываемого и экспериментального кода и драйверов. Без этого не появятся некоторые важные пункты:

Листинг 3.2: Выбор экспериментальных кодов/драйверов, основные параметры

```
Code maturity level options --->
  [*] Prompt for development and/or incomplete code/drivers"
General setup  --->
  [*] Support for hot-pluggable devices
```

Удостоверьтесь, что каждый драйвер, жизненно необходимый для загрузки системы (например, SCSI-контроллера), собран *внутри* ядра, а не как модуль, иначе система совсем не сможет загрузиться.

Далее выберите нужное семейство процессоров:

Листинг 3.3: Выбор подходящего семейства процессоров

```
Processor type and features --->
  (измените в соответствии со своей системой)
  (Athlon/Duron/K7) Processor family
```

Теперь войдите в `File Systems` и выберите поддержку всех нужных файловых систем. *Не* компилируйте их как модули, иначе система Gentoo не сможет смонтировать дисковые разделы. Также выберите `Virtual memory` и `/proc file system`. При использовании ядра 2.4, вам потребуется включить `/dev file system`, т.к. это ядро не поддерживает `udev`.

Листинг 3.4: Выбор необходимых файловых систем

```
(при использовани ядра 2.4.x)
File systems --->
  [*] Virtual memory file system support (former shm fs)
  [*] /proc file system support
  [*] /dev file system support (EXPERIMENTAL)
  [*]   automatically mount /dev at boot
  [ ] /dev/pts file system for Unix98 PTYs

(при использовании ядра 2.6.x)
File systems --->
  Pseudo Filesystems --->
    [*] /proc file system support
    [*] Virtual memory file system support (former shm fs)

(включите поддержку всех нужных вам файловых систем)
  <*> Reiserfs support
  <*> Ext3 journalling file system support
  <*> JFS filesystem support
  <*> Second extended fs support
  <*> XFS filesystem support
```

Если ваш BIOS не поддерживает жесткие диски большого размера, и вы ограничили перемычками сообщаемый диском объем, то для получения доступа ко всему диску нужно включить следующие параметры ядра:

Листинг 3.5: Включение поддержки автокоррекции геометрии

```
(только в ядре 2.4.x)
ATA/IDE/MFM/RLL support --->
  IDE, ATA and ATAPI Block devices --->
    <*>   Include IDE/ATA-2 DISK support
    [ ]     Use multi-mode by default
    [*]     Auto-Geometry Resizing support
```

Не забудьте включить поддержку режима DMA для ваших дисков:

Листинг 3.6: Включение DMA

```
Device Drivers --->
  ATA/ATAPI/MFM/RLL support --->
    [*] Generic PCI bus-master DMA support
    [*]   Use PCI DMA by default when available
```

Если вы используете PPPoE или коммутируемого соединения для соединения с интернетом, потребуется включить в ядре следующие функции:

Листинг 3.7: Выбор нужных драйверов PPPoE

```
(при использовании ядра 2.4.x)
```

```
Network device support --->
  <*> PPP (point-to-point protocol) support
  <*>   PPP support for async serial ports
  <*>   PPP support for sync tty ports

(при использовании ядра 2.6.x)
Device Drivers --->
  Networking support --->
    <*> PPP (point-to-point protocol) support
    <*>   PPP support for async serial ports
    <*>   PPP support for sync tty ports
```

Два параметра сжатия данных не являются обязательными, но и не повредят. То же относится и к параметру `PPP over Ethernet` который нужен только для `rp-pppoe`, для поддержки PPPoE в режиме ядра.

При необходимости, не забудьте включить в ядре поддержку своей сетевой платы.

Если у вас процессор Intel с поддержкой HyperTreading или многопроцессорная система, следует включить «симметричную многопроцессорную обработку»:

Листинг 3.8: Включение поддержки SMP

```
Processor type and features  --->
  [*] Symmetric multi-processing support
```

Если вы пользуетесь устройствами ввода на шине USB, (например, мышью или клавиатурой), не забудьте включить их поддержку:

Листинг 3.9: Включение поддержки USB для устройств ввода

```
USB Support --->
  <*>   USB Human Interface Device (full HID) support
```

Владельцам ноутбуков, которым нужна поддержка PCMCIA, *не нужно* включать драйверы PCMCIA в ядре, выбрав ядро версии 2.4. В пакете `pcmcia-cs`, который будет устанавливаться позже, есть более свежие драйверы. Пользователям ядра 2.6 следует использовать драйверы PCMCIA, включенные в ядро.

Наряду с включением поддержки PCMCIA в ядре 2.6, не забудьте включить поддержку моста PCMCIA своей системы:

Листинг 3.10: Включение поддержки PCMCIA для ядра 2.6

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->
  PCCARD (PCMCIA/CardBus) support  --->
    <*> PCCard (PCMCIA/CardBus) support
(если нужно (обычно нужно), включите поддержку старых 16-битных карт PCMCIA)
    <*>   16-bit PCMCIA support
    [*]   32-bit CardBus support
(выберите наиболее подходящий мост)
    --- PC-card bridges
    <*> CardBus yenta-compatible bridge support (NEW)
    <*> Cirrus PD6729 compatible bridge support (NEW)
    <*> i82092 compatible bridge support (NEW)
    <*> i82365 compatible bridge support (NEW)
    <*> Databook TCIC host bridge support (NEW)
```

Закончив настройку ядра, приступайте к компиляции и установке.

Компиляция и установка

Теперь, когда ядро настроено, настало время его скомпилировать и установить. Выйдите из настройки и запустите процесс компиляции:

Листинг 3.11: Компиляция ядра

```
(для ядра 2.4)
# make dep && make bzImage modules modules_install

(для ядра 2.6)
# make && make modules_install
```

По завершении компиляции, скопируйте образ ядра в каталог `/boot`. Используйте любое имя, которое вам покажется подходящим для избранного ядра, и запомните его, так как позже при настройке начального загрузчика оно потребуется. Не забудьте заменить `<версия-ядра>` на имя и версию своего ядра.

Листинг 3.12: Установка ядра

```
# cp arch/i386/boot/bzImage /boot/<версия-ядра>
```

Теперь переходите к модулям ядра.

7.d. Альтернатива: использование genkernel

Если вы читаете эти строки, значит, вы собираетесь использовать наш сценарий `genkernel` для настройки ядра за вас.

Теперь когда исходные коды ядра установлены, настало время скомпилировать ядро, собрав его автоматически при помощи сценария `genkernel`. `genkernel` выполняет настройку ядра практически так же, как настраивается ядро на установочном компакт-диске. То есть система, ядро которой собрано `genkernel`, как правило, будет выполнять определение всех устройств при загрузке. Поскольку genkernel не требует никакой ручной настройки ядра, это идеальное решение для тех, кому неуютно от одной мысли, что придется компилировать ядро.

Рассмотрим, как использовать genkernel. Сначала установите его:

Листинг 4.1: Установка genkernel

```
# emerge genkernel
```

Теперь, если вы собираетесь использовать ядро ветви 2.6, скопируйте настройку ядра, используемую установочным диском, в каталог, где genkernel ищет настройку по умолчанию:

Листинг 4.2: Копирование файла настройки ядра с установочного диска

```
(только если вы собираетесь настраивать ядро 2.6.x)
# zcat /proc/config.gz > /usr/share/genkernel/x86/kernel-config-2.6
```

Теперь скомпилируйте исходный код ядра, запустив `genkernel all`. Имейте в виду, что компиляция займет приличное время, поскольку `genkernel` собирает ядро, поддерживающее практически любое оборудование.

Если в загрузочном разделе не используется файловая система ext2 или ext3, то вам придется вручную настроить ядро, запустив `genkernel --menuconfig all`, добавив поддержку используемой файловой системы *в* ядро (т.е. *не* модулем). Пользователям EVMS2 или LVM2, вероятно, захочется также добавить параметр командной строки `--evms2` или `--lvm2`.

Листинг 4.3: Запуск genkernel

```
# genkernel all
```

В итоге работы `genkernel` будет создано ядро, полный набор модулей и *начальный корневой диск* (initrd). Ядро и initrd будут использоваться в процессе настройки загрузчика системы, которая описана далее в руководстве. Запишите имена ядра и начального корневого диска, так как они понадобятся при составлении конфигурационного файла загрузчика. initrd запускается сразу при начальной загрузке, выполняя автоматическое определение устройств (точно так же, как при запуске установочного диска), до

запуска «настоящей» системы.

Листинг 4.4: Уточнение названия созданного образа ядра и initrd

```
# ls /boot/kernel* /boot/initramfs*
```

Теперь давайте еще на шаг приблизим нашу систему к установочному диску: соберем `coldplug`. Если initrd определяет оборудование, необходимое для загрузки системы, `coldplug` выполняет автоопределение всех остальных устройств. Для установки и подключения `coldplug` введите команду:

Листинг 4.5: Установка и подключение coldplug

```
# emerge coldplug
# rc-update add coldplug boot
```

7.e. Модули ядра

Настройка модулей

Модули, которые требуется загружать автоматически, нужно указать в `/etc/modules.autoload.d/kernel-2.4` (или `kernel-2.6`). Также, при желании, модулям можно сообщить дополнительные параметры.

Для просмотра всех доступных модулей запустите команду `find`, заменив «<версия ядра>» на версию только что собранного ядра:

Листинг 5.1: Просмотр перечня доступных модулей

```
# find /lib/modules/<kernel version>/ -type f -iname '*.o' -or -iname '*.ko'
```

Например, для автоматической загрузки модуля `3c59x.o` измените файл `kernel-2.4` или `kernel-2.6`, указав в нем имя модуля.

Листинг 5.2: Изменение /etc/modules.autoload.d/kernel-2.4

```
(показан пример для ядра 2.4.x)
# nano -w /etc/modules.autoload.d/kernel-2.4
```

Листинг 5.3: /etc/modules.autoload.d/kernel-2.4 или kernel-2.6

```
3c59x
```

Теперь переходите к настройке параметров системы.

8. Настройка параметров системы

8.a. Параметры файловых систем

Что такое fstab?

В Linux все разделы, используемые системой, должны быть перечислены в `/etc/fstab`. В этом файле указываются точки подключения разделов (mountpoints, местоположение разделов в файловой системе), порядок подключения, а также дополнительные параметры (автоматический или ручной режим подключения, достаточность прав пользователя для подключения и т.п.)

Создание /etc/fstab

В `/etc/fstab` используется специальный формат. Каждая строка состоит из шести полей, разделяемых пробелами, знаками табуляции или их сочетанием. Каждое поле имеет свое назначение:

- Первое поле обозначает **раздел (partition)** (путь к файлу устройства).
- Второе поле указывает **точку подключения (mountpoint)**, в которую монтируется раздел.
- Третье поле задает тип **файловой системы (filesystem)**, используемой в разделе.
- В четвертом поле указываются **параметры подключения (mountoptions)**, используемые `mount` при подключении раздела. Поскольку для каждой файловой системы существуют свои параметры, рекомендуется прочитать страницу справки по mount (`man mount`), где приведен их полный перечень. При указании нескольких параметров подключения их следует разделять запятыми.
- Пятое поле используется `dump` для определения, требуется ли резервное копирование раздела средствами dump. Обычно это поле можно просто установить в `0` (ноль).
- Шестое поле используется `fsck` для определения порядка **проверки (check)** файловых систем после некорректного завершения работы системы. Для корневой файловой системы рекомендуется значение `1`, а для остальных — `2` (или `0`, когда проверка файловой системы не требуется).

Важно: Вариант файла `/etc/fstab` по умолчанию, входящий в Gentoo, *не является работоспособным*. Вам **потребуется создать** свой собственный `/etc/fstab`.

Листинг 1.1: Открытие /etc/fstab

```
# nano -w /etc/fstab
```

Укажите правила, соответствующие вашей схеме разбивки, и добавьте правила для `/proc`, для `tmpfs`, для своих дисководов CD-ROM (если есть другие разделы или устройства, их тоже можно указать).

Теперь на основе приведенного *примера* создайте собственный файл `/etc/fstab`:

Параметр `auto` позволяет `mount` определять тип файловой системы автоматически (рекомендуется для съемных носителей, которые могут оказаться размечены в одной из множества существующих файловых систем), а `user` позволяет монтировать компакт-диски обычным пользователям.

Чтобы повысить быстродействие, большинству пользователей стоит добавить параметр `noatime` в параметры подключения, что приведет к ускорению за счет отключения регистрации отметки времени доступа к файлам (обычно в ней все равно нет необходимости):

Перепроверьте свой файл `/etc/fstab`, сохраните его, и выйдите из редактора, чтобы продолжить настройку.

8.b. Параметры сети

Hostname, Domainname и т. д.

Еще один вопрос, который нужно решить пользователю — как назвать свой компьютер. Он кажется довольно простым, но *многие* затрудняются дать подходящее имя для своей Linux-системы. Чтобы вам стало легче, запомните, что какое бы имя вы не выбрали, потом его всегда можно изменить. Например, вы могли бы просто назвать свою систему `tux`, а домен — `homenetwork`.

Листинг 2.1: Установка имени узла

```
# nano -w /etc/conf.d/hostname

(присвойте переменной HOSTNAME имя своего узла)
HOSTNAME="tux"
```

Во-вторых, установим имя домена (domainname) в `/etc/conf.d/net`:

Листинг 2.2: Установка имени домена

```
# nano -w /etc/conf.d/net

(присвойте переменной DNSDOMAIN имя своего домена)
dns_domain_lo="homenetwork"
```

Если у вас есть домен NIS (а если вы не знаете, что это такое, то у вас его точно нет), его также необходимо указать:

Листинг 2.3: Установка имени NIS-домена

```
# nano -w /etc/conf.d/net

(укажите название своего домена NIS переменной nis_domain)
nis_domain_lo="my-nisdomain"
```

Настройка сети

Прежде, чем возмутиться: «Эй, мы же все это уже делали!» — вспомните, что подключение к сети, настроенное вначале, было предназначено лишь для установки Gentoo. Теперь же вы настраиваете сеть для постоянного использования.

Примечание: Более подробные сведения о сетях, включая дополнительные темы, такие как объединение, образование мостов, настройка виртуальных сетей (VLAN) 802.1Q или беспроводных сетей, представлены в разделе настройка сети в Gentoo.

Все настройки сети собраны в файле `/etc/conf.d/net`. В нем используется простой формат, хотя, если вы не знакомы с ручной настройкой сети, он не слишком очевиден. Но не бойтесь, мы все объясним. В файле `/etc/conf.d/net.example` приведен подробно прокомментированный пример, охватывающий много различных конфигураций.

По умолчанию используется DHCP. Чтобы DHCP заработал, требуется установить DHCP-клиент, как описано далее в разделе Установка нужных системных средств. Не забудьте установить DHCP-клиент.

Если настройка сетевого подключения нужна вам для указания специфических параметров DHCP, или из-за того, что вы вообще не используете DHCP, откройте `/etc/conf.d/net` в своем любимом редакторе (в этом примере использован `nano`):

Листинг 2.4: Открытие /etc/conf.d/net для изменения

```
# nano -w /etc/conf.d/net
```

Вы увидите следующее:

Листинг 2.5: /etc/conf.d/net по умолчанию

```
# This blank configuration will automatically use DHCP for any net.*
# scripts in /etc/init.d.  To create a more complete configuration,
# please review /etc/conf.d/net.example and save your configuration
# in /etc/conf.d/net (this file :]!).

(# Этот пустой файл настройки приводит к автоматическому использованию
 # DHCP всеми сценариями net.* из /etc/init.d. Для создания более полной
 # настройки, пожалуйста, просмотрите /etc/conf.d/net.example, а свою
 # настройку сохраните в /etc/conf.d/net (в этом файле :]!).           )
```

Чтобы указать свой собственный адрес IP, маску сети и шлюз, потребуется настроить как `config_eth0`, так и `routes_eth0`:

Листинг 2.6: Ручная настройка параметров IP для eth0

```
config_eth0=( "192.168.0.2 netmask 255.255.255.0 brd 192.168.0.255" )
routes_eth0=( "default gw 192.168.0.1" )
```

Чтобы при использовании DHCP указать специфические параметры, определите `config_eth0` и `dhcp_eth0`:

Листинг 2.7: Автоматическое получение адреса IP для eth0

```
config_eth0=( "dhcp" )
dhcp_eth0="nodns nontp nonis"
```

Список допустимых параметров дан в файле `/etc/conf.d/net.example`.

Если у вас несколько сетевых интерфейсов, повторите эти шаги для `config_eth1`, `config_eth2` и т.д.

Теперь сохраните параметры и выйдите из редактора, чтобы продолжить настройку.

Автоматический запуск сетевого подключения при загрузке

Для запуска сетевых интерфейсов при загрузке необходимо добавить их в уровень запуска по умолчанию. Если у вас интерфейсы типа PCMCIA, пропустите этот шаг, поскольку интерфейсы PCMCIA запускаются сценарием инициализации PCMCIA.

Листинг 2.8: Добавление net.eth0 в уровень запуска default

```
# rc-update add net.eth0 default
```

Если у вас несколько сетевых интерфейсов, потребуется создать для них соответствующие сценарии инициализации `net.eth1`, `net.eth2` и т.д. Для этого можно использовать `ln`:

Листинг 2.9: Создание дополнительных сценариев инициализации

```
# cd /etc/init.d
# ln -s net.lo net.eth1
# rc-update add net.eth1 default
```

Указание сетевых узлов

Теперь расскажем системе Linux о вашей сети. Эти сведения указываются в `/etc/hosts`, и помогают разрешению имен в IP-адреса для узлов, не обрабатываемых сервером имен. Требуется определить вашу систему. Также можно определить другие системы в сети, если вы не собираетесь устанавливать собственную систему DNS.

Листинг 2.10: Открытие /etc/hosts

```
# nano -w /etc/hosts
```

Листинг 2.11: Указание сведений об узлах сети

```
(определение текущей системы)
127.0.0.1     tux.homenetwork tux localhost

(определите другие машины в своей сети,
для этого у них должен быть статический IP-адрес.)

192.168.0.5   jenny.homenetwork jenny
192.168.0.6   benny.homenetwork benny
```

Чтобы продолжить настройку, сохраните файл и выйдите из редактора.

8.c. Параметры системы

Пароль root

Прежде всего, нужно установить пароль root (администратора), набрав:

Листинг 3.1: Установка пароля root

```
# passwd
```

Если вы хотите, чтобы root мог входить в систему через последовательный терминал, добавьте `tts/0` в `/etc/securetty`:

Листинг 3.2: Добавление tts/0 to /etc/securetty

```
# echo "tts/0" >> /etc/securetty
```

Параметры системы

Для общей настройки системы в Gentoo используется `/etc/rc.conf`. Откройте `/etc/rc.conf` и с удовольствием прочитайте все комментарии, находящиеся в этом файле :)

Листинг 3.3: Открытие /etc/rc.conf

```
# nano -w /etc/rc.conf
```

Завершив изменение `/etc/rc.conf` сохраните файл и выйдите из редактора.

Как видите, этот файл подробно прокомментирован, что поможет вам в настройке необходимых конфигурационных переменных. Можно настроить систему на использование unicode, а также указать редактор по умолчанию и диспетчер отображения (например, gdm или kdm).

Для управления раскладками клавиатуры в Gentoo используется `/etc/conf.d/keymaps`. Для настройки своей клавиатуры измените его.

Листинг 3.4: Открытие /etc/conf.d/keymaps

```
# nano -w /etc/conf.d/keymaps
```

Будьте особенно тщательны при установке переменной раскладки клавиатуры `(KEYMAP)`: выбрав неверную раскладку, вы можете получить непредсказуемый результат при попытке ввода с клавиатуры.

По завершении изменения `/etc/conf.d/keymaps` сохраните файл и выйдите из редактора.

Для настройки параметров часов в Gentoo используется `/etc/conf.d/clock`. Изменяйте его в соответствии со своими потребностями.

Если аппаратные часы вашей системы настроены не на часовой пояс UTC (Гринвич), в файл необходимо добавить строку `CLOCK="local"`. В противном случае вы заметите сдвиг часового пояса.

После завершения настройки `/etc/conf.d/clock` сохраните файл и выйдите из редактора.

Переходите к установке нужных системных средств.

9. Установка нужных системных средств

9.a. Диспетчер устройств

Если вы используете ядро 2.4, *и при этом* устанавливаете Gentoo из файла третьей стадии (stage3), вам *необходимо* еще кое-что сделать. Так как теперь в Gentoo по умолчанию используется udev, а ядром 2.4 udev не поддерживается, вам потребуется установить devfsd, и убрать udev.

Листинг 1.1: Установка devfsd

```
(для тех, кто использует ядро 2.4.x при установке с третьей стадии)
# emerge --unmerge udev
# emerge devfsd
```

9.b. Системные службы журналирования

Некоторые средства не включены в архив *stage3*, поскольку одинаковые возможности можно обеспечить различными пакетами. Теперь вы сами выберете, какие именно установить.

Первый инструмент, который вам необходимо выбрать, должен дать системе возможность журналирования. У Unix и Linux превосходная история развития журналирования — при желании в файлах журналов можно регистрировать любой чих, происходящий в вашей системе. Это обеспечивается *системной службой журналирования*.

В Gentoo предлагается несколько служб журналирования на выбор. Это `sysklogd`, традиционный набор журналирующих демонов, `syslog-ng`, расширенная служба журналирования, и `metalog` — служба журналирования с очень гибкими возможностями настройки. Возможно, в Portage имеются и другие службы журналирования: количество доступных пакетов растет день ото дня.

Если вы планируете использовать `sysklogd` или `syslog-ng`, позднее может потребоваться установка `logrotate`, поскольку в этих службах журналирования не предусмотрен никакой механизм ротации системных журналов.

Чтобы установить выбранную службу журналирования, воспользуйтесь `emerge`, а затем добавьте ее в уровень запуска по умолчанию с помощью `rc-update`. В следующем примере показана установка `syslog-ng`. Вместо нее вы можете подставить другую службу журналирования:

Листинг 2.1: Установка системной службы журналирования

```
# emerge syslog-ng
# rc-update add syslog-ng default
```

9.c. Дополнительно: демон cron

Следующий демон — cron. Хотя он является дополнительным, и не обязателен для работы вашей системы, будет благоразумным установить его. Но что же такое демон cron? Демон cron выполняет команды по расписанию. Это очень удобно, когда нужно выполнять какие-либо команды регулярно (например, ежедневно, еженедельно или ежемесячно).

В Gentoo предлагаются три варианта демона cron на выбор: `dcron`, `fcron` и `vixie-cron`. Установка любого из них подобна установке системной службы журналирования. Однако, при установке `dcron` или `fcron` для настройки требуется выполнение дополнительной команды, а именно `crontab /etc/crontab` . Если вы не знаете, что выбрать, используйте `vixie-cron`.

При бессетевой установке доступен только `vixie-cron`. Если вам нужен другой демон cron, его можно установить позже.

Листинг 3.1: Установка демона cron

```
# emerge vixie-cron
# rc-update add vixie-cron default
(только если выбран dcron или fcron) # crontab /etc/crontab
```

9.d. Дополнительно: индексация файлов

Если вы хотите индексировать файлы в своей системе, чтобы быстро находить их с помощью `locate`, нужно установить `sys-apps/slocate`:

Листинг 4.1: Установка slocate

```
# emerge slocate
```

9.e. Утилиты для обслуживания файловых систем

Для проверки целостности файловых систем, создания дополнительных файловых систем, и т.п., вам потребуются определенные утилиты, состав которых зависит от используемых файловых систем.

В следующей таблице перечислены утилиты, которые необходимо устанавливать для обслуживания файловых систем различных типов:

Файловая система	Утилиты	Команда установки
XFS	xfsprogs	`emerge xfsprogs`
ReiserFS	reiserfsprogs	`emerge reiserfsprogs`
JFS	jfsutils	`emerge jfsutils`

Если вы используете EVMS, также необходимо установить `evms`:

Листинг 5.1: Установка утилит EVMS

```
# USE="-gtk" emerge evms
```

`USE="-gtk"` предотвратит установку пакетов, от которых зависит данный. При желании включить графические средства `evms`, потом можно перекомпилировать `evms`.

Если вам не нужны никакие дополнительные сетевые средства (типа rp-pppoe или клиента dhcp), переходите к настройке начального загрузчика.

9.f. Сетевые средства

Дополнительно: установка клиента DHCP

Если требуется, чтобы Gentoo автоматически получала IP-адрес для ваших сетевых интерфейсов, необходимо установить `dhcpcd` (или любой другой клиент DHCP, список имеющихся клиентов DHCP см. в главе Модульное построение сети). Если не сделать этого сейчас, то после завершения установки вы не сможете подключиться к интернету!

Листинг 6.1: Установка dhcpcd

```
# emerge dhcpcd
```

Дополнительно: установка клиента PPPoE

Если для выхода в сеть требуется `rp-pppoe`, его нужно установить.

Листинг 6.2: Установка rp-pppoe

```
# USE="-X" emerge rp-pppoe
```

`USE="-X"` запрещает установку xorg-x11 в порядке зависимости (в `rp-pppoe` есть графические средства; если их нужно подключить, можно перекомпилировать `rp-pppoe` позже, или же установить xorg-x11 сейчас: при его установке потребуется много времени на компиляцию).

Дополнительно: утилиты RAID для оборудования IBM

Если в составе системы на базе POWER5 вы используете RAID-массивы SCSI, стоит задуматься об установке `iprutils`, которые, среди прочего, позволяют работать с дисковыми массивами, выяснять состояние дисков в составе массива и обновлять микрокод.

Листинг 6.3: Установка iprutils

```
# emerge iprutils
```

Теперь переходите к настройке начального загрузчика.

10. Настройка начального загрузчика

10.a. Выбор загрузчика

Введение

Теперь, когда ядро настроено и собрано, а нужные конфигурационные файлы заполнены как надо, пришло время установить программу, которая будет запускать ваше ядро при старте системы. Такого рода программа называется *загрузчиком*. Для архитектуры x86 в Gentoo Linux есть загрузчики GRUB и LILO. Но прежде, чем приступить к установке одного из двух загрузчиков, мы расскажем, как настроить кадровый буфер (естественно, если вы захотите). С помощью кадрового буфера можно работать в командной строке Linux на фоне графических элементов (например, симпатичного изображения из загрузочной заставки Gentoo).

Дополнительно: кадровый буфер

Если вы настроили в ядре поддержку кадрового буфера (или оставили настройки по умолчанию при использовании `genkernel`), вы можете включить буфер, добавив параметр `vga` и/или `video` в файл конфигурации своего загрузчика.

Для начала, вам надо узнать тип используемого кадрового буфера. При использовании исходных кодов ядра, доработанных для Gentoo (таких как `gentoo-sources`), у вас есть возможность выбрать `vesafb-tng` в качестве `типа драйвера VESA` (в этих исходных кодах ядра он используется по умолчанию). При использовании `vesafb-tng` параметр `vga` указывать не нужно. В других случаях используется драйвер `vesafb`, а параметр `vga` необходимо указывать.

Параметр `vga` устанавливает разрешение и глубину цвета, используемые кадровым буфером `vesafb`. Как отмечено в `/usr/src/linux/Documentation/fb/vesafb.txt` (который устанавливается в составе пакета с деревом исходных кодов ядра), кадровому буферу необходимо передавать код режима VESA, соответствующий нужному разрешению и цветности.

В следующей таблице приведены коды режимов для указания в параметре `vga`, а также соответствующие им значения разрешения и цветности.

	640x480	800x600	1024x768	1280x1024
256	0x301	0x303	0x305	0x307
32 тыс	0x310	0x313	0x316	0x319
64 тыс	0x311	0x314	0x317	0x31A
16 млн	0x312	0x315	0x318	0x31B

Параметр `video` отвечает за порядок отображения кадрового буфера. В нем указывается название драйвера кадрового буфера (`vesafb` для ядра 2.6 или `vesa` для ядра 2.4), а затем режимы, которые нужно включить. Все режимы приведены в `/usr/src/linux/Documentation/fb/vesafb.txt`, а здесь мы расскажем о трех самых используемых:

Переменная	Описание
ywrap	**считать, что видеоплата может закольцовывать свою память (продолжать выборку с начального адреса, когда достигнут конечный)**
mtrr:n	**установка регистров MTRR; допустимые значения `n`: 0 - отключено 1 - без кэширования 2 - отложенная запись (write-back) 3 - объединенная запись (write-combining) 4 - сквозная запись (write-through)**
`mode`	**(только для `vesafb-tng`) Установить разрешение, цветность и частоту кадров. Например, `1024x768-32@85` для разрешения 1024x768, 32-битного цвета и частоты кадров 85 Гц.**

В результате должно получиться что-то вроде `vga=0x318 video=vesafb:mtrr:3,ywrap` или `video=vesafb:mtrr:3,ywrap,1024x768-32@85`. Запомните (или запишите) составленное значение, скоро оно понадобится.

Перейдем к установке GRUB *или* LILO.

10.b. По умолчанию: использование GRUB

Введение в терминологию GRUB

Самое сложное в освоении GRUB — освоиться с тем, как в нем именуются жесткие диски и разделы. Ваш Linux-раздел `/dev/hda1`, скорее всего, в GRUB станет называться `(hd0,0)`. Обратите внимание на круглые скобки вокруг `hd0,0` — они обязательны.

Жесткие диски нумеруются, начиная с нуля, а не с «a»; разделы — с нуля, а не с единицы. Нужно помнить, что в виде hd нумеруются только жесткие диски, но не устройства atapi-ide, такие как приводы компакт-дисков. Та же нумерация используется для устройств SCSI (обычно им присваиваются номера большие, чем устройствам IDE, кроме случаев, когда BIOS настроен на загрузку с устройства SCSI). Когда BIOS настроен на загрузку с другого жесткого диска (например, с первичного ведомого), *именно этот* жесткий диск и становится `hd0`.

Например, если у вас есть жесткий диск `/dev/hda`, привод CD-ROM `/dev/hdb`, записывающий CD `/dev/hdc`, второй жесткий диск `/dev/hdd`, а устройств SCSI нет, то разделу `/dev/hdd7` будет соответствовать `(hd1,6)`. Возможно, это покажется запутанным (так и есть), но, как мы увидим, в GRUB есть механизм автодополнения по tab, облегчающий жизнь обладателям множества жестких дисков и разделов, а также тем, кто теряется в схеме нумерации устройств GRUB.

Почувствовав, что к чему, пора установить GRUB.

Установка GRUB

Для установки GRUB сначала добавим его в систему:

Листинг 2.1: Установка GRUB

```
# emerge grub
```

Хотя GRUB уже установлен, нам еще потребуется подправить его файл конфигурации, и поместить GRUB в MBR, чтобы он автоматически загружал ядро. С помощью `nano` (или другого редактора) создайте `/boot/grub/grub.conf`:

Листинг 2.2: Создание /boot/grub/grub.conf

```
# nano -w /boot/grub/grub.conf
```

Теперь заполним `grub.conf` своими значениями. Ниже приведены два варианта `grub.conf` для показанного примера разбиения дисков. Первый вариант `grub.conf` мы подробно прокомментировали. Удостоверьтесь, что у себя вы указываете имя *своего* файла образа ядра, и при необходимости имя *своего* образа начального корневого диска (initrd).

- первый вариант `grub.conf` — для тех, кто при сборке ядра обходился без `genkernel`
- второй вариант `grub.conf` — для тех, кто при сборке ядра пользовался `genkernel`

Примечание: Если ваша корневая файловая система — JFS, *необходимо* добавить «ro» в строку `kernel`, поскольку JFS «накатывает» свой журнал перед тем, как разрешить монтирование раздела на чтение-запись.

Листинг 2.3: grub.conf для тех, кто обошелся без genkernel

```
# какой пункт загружать по умолчанию: 0 - первый, 1 - второй и т.д.
default 0
# сколько секунд ждать до начала загрузки пункта по умолчанию
timeout 30
# симпатичная заставка, добавить по вкусу :)
# закомментируйте, если у вас не установлена графическая видеоплата
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title=Gentoo Linux 2.6.12-r10
# раздел с файлом образа ядра (или операционной системой)
root (hd0,0)
kernel /boot/kernel-2.6.12-gentoo-r10 root=/dev/hda3

# следующие четыре строки нужны только для двойной загрузки с Windows
```

```
# в этом примере Windows находится на /dev/hda6
title=Windows XP
rootnoverify (hd0,5)
makeactive
chainloader +1
```

Листинг 2.4: grub.conf для тех, кто пользовался genkernel

```
default 0
timeout 30
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title=Gentoo Linux 2.6.12-r10
root (hd0,0)
kernel /boot/kernel-genkernel-x86-2.6.12-gentoo-r10 root=/dev/ram0 init=/linuxrc ramdisk=8192
initrd /boot/initramfs-genkernel-x86-2.6.12-gentoo-r10

# нужно только для двойной загрузки
title=Windows XP
rootnoverify (hd0,5)
makeactive
chainloader +1
```

Примечание: Параметр `udev`, указанный в конце строки kernel, необходим для обхода ошибки в некоторых версиях genkernel, *если* вы вообще используете udev (по умолчанию — используется).

Примечание: Если вы разбили жесткий диск по-другому, или у вас другое ядро, внесите необходимые изменения. При этом убедитесь, что все пути, следующие за упоминанием устройства GRUB (например `(hd0,0)`), приведены относительно точки подключения, а не корня файловой системы. Другими словами, `(hd0,0)/grub/splash.xpm.gz` — на самом деле `/boot/grub/splash.xpm.gz`, так как `(hd0,0)` — это `/boot`.

Кроме того, если вы избрали другую схему разбиения диска, и не выделяли для `/boot` отдельный раздел, префикс `/boot`, использованный в примерах выше, *обязателен*. Если же вы следовали рекомендованному нами плану разбиения, префикс `/boot` не требуется, но все работает благодаря символьной ссылке `boot`. Короче говоря, приведенные примеры должны работать независимо от того, есть у вас отдельный раздел для `/boot` или нет.

Если вам надо передать ядру дополнительные параметры, просто добавьте их в конец строки kernel. Один параметр мы уже передаем ядру (`root=/dev/hda3` или `real_root=/dev/hda3`); можно добавлять и другие, например, такие как параметры `video` и/или `vga` для кадрового буфера, обсуждавшиеся выше.

Если вы используете ядро 2.6.7 или выше, а объем жесткого диска ограничили перемычками из-за того, что BIOS не в состоянии работать с дисками большого размера, вам потребуется добавить `hdx=stroke`.

Тем, кто использует `genkernel`, нужно помнить, что их ядро использует такие же загрузочные параметры, как на установочном компакт-диске. Например, если у вас есть устройства SCSI, следует передать ядру параметр `doscsi`.

Теперь сохраните `grub.conf` и выйдите из редактора. Вам по-прежнему необходимо записать GRUB в MBR (Master Boot Record), чтобы GRUB автоматически запускался при загрузке системы.

Разработчики GRUB рекомендуют использовать `grub-install`. Однако, на случай некорректной работы `grub-install` есть возможность записать GRUB вручную.

Переходите к разделу по умолчанию: установка GRUB с помощью grub-install или альтернатива: установка GRUB вручную.

По умолчанию: установка GRUB с помощью grub-install

Для установки GRUB вам надо выполнить команду `grub-install`. Однако, `grub-install` не заработает сам по себе, т.к. мы находимся в среде с измененным корневым каталогом. Нам нужно создать файл `/etc/mtab`, перечислив в нем все смонтированные файловые системы. К счастью, для этого есть очень легкий способ: просто скопируйте содержимое `/proc/mounts` поверх `/etc/mtab`, исключив строку `rootfs` , если вы не создавали отдельный загрузочный раздел. Следующая команда подойдет в обоих случаях:

Листинг 2.5: Создание /etc/mtab

```
# grep -v rootfs /proc/mounts > /etc/mtab
```

Теперь мы можем установить GRUB, используя `grub-install`:

Листинг 2.6: Выполнение grub-install

```
# grub-install /dev/hda
```

Если у вас есть вопросы о GRUB, пожалуйста, обратитесь к GRUB FAQ (англ.) или руководству по GRUB (англ.).

Переходите к перезагрузке системы.

Альтернатива: установка GRUB вручную

Для начала настройки, введите `grub`. Вы увидите приглашение `grub>` — это командная строка grub. Теперь потребуется набрать команды, нужные для установки загрузочной записи GRUB на ваш жесткий диск.

Листинг 2.7: Запуск оболочки GRUB

```
# grub
```

Примечание: Если у вас нет приводов для дискет, к приведенной команде добавьте `--no-floppy`, чтобы grub зря не опрашивал несуществующие дисководы.

В приводимом примере мы хотим установить GRUB так, чтобы он считывал нужную информацию с загрузочного раздела `/dev/hda1`, а загрузочная запись GRUB находилась в MBR (Master Boot Record) жесткого диска, чтобы первое, что мы видели после включения компьютера — это приглашение GRUB. Естественно, если вы при установке отклонялись от предлагаемой схемы, внесите необходимые поправки.

Находясь в GRUB, можно использовать автодополнение по клавише TAB. К примеру, если ввести «`root (`», а затем TAB, появится список устройств (таких как `hd0`). Если ввести «`root (hd0,`» и нажать TAB, появится список для выбора раздела из возможных (такого как `hd0,0`).

Благодаря автодополнению установка GRUB не так сложна. Теперь приступим к настройке GRUB.

Листинг 2.8: Установка GRUB в MBR

```
grub> root (hd0,0)          (указание расположения раздела с /boot)
grub> setup (hd0)           (установка GRUB в MBR)
grub> quit                  (выход из оболочки GRUB)
```

Примечание: Если вы хотите установить GRUB в определенный раздел вместо MBR, команду `setup` потребуется исправить так, чтобы она указывала на нужный раздел. Например, команда для установки GRUB в `/dev/hda3` — `setup (hd0,2)`. Однако, так поступают немногие.

С дополнительными вопросами о GRUB, пожалуйста, обращайтесь к GRUB FAQ (англ.) или руководству по GRUB (англ.).

Примечание: В случае переустановки ядра в будущем, вам больше не потребуется копировать содержимое файлов. После компиляции ядра просто запускайте `make install`: копирование файлов и изменение конфигурации GRUB произойдет автоматически.

Переходите к перезагрузке системы.

10.c. Альтернатива: использование LILO

Установка LILO

LILO (сокращение от LInux LOader) — это проверенная временем рабочая лошадка среди загрузчиков Linux-систем. Но ей недостает ряда возможностей, которые есть в GRUB (и в том числе в этом заключается причина растущей популярности GRUB). LILO все еще используется, потому что на некоторых системах он работает, а GRUB — нет. Конечно же, он используется еще и потому, что многие просто знакомы с LILO и сроднились с ним. Так или иначе, в Gentoo поддерживаются оба загрузчика, и вы, видимо, решили использовать LILO.

Установка LILO в систему проста как пробка: просто используйте `emerge`.

Листинг 3.1: Установка LILO

```
# emerge lilo
```

Настройка LILO

Для настройки LILO нужно создать файл `/etc/lilo.conf`. Запустите свой любимый редактор (в руководстве мы для единообразия используем `nano`) и создайте файл.

Листинг 3.2: Создание /etc/lilo.conf

```
# nano -w /etc/lilo.conf
```

Несколькими разделами раньше мы попросили вас запомнить название созданного файла образа ядра. В следующем примере используется предложенная нами схема разделения диска. Пример разделен на две части:

- одна — для тех, кто не пользовался для сборки ядра `genkernel`
- другая — для тех, кто при сборке ядра пользовался `genkernel`

Удостоверьтесь, что у себя вы указываете имя *своего* файла образа ядра, и при необходимости имя *своего* образа начального корневого диска (initrd).

Примечание: Если ваша корневая файловая система — JFS, *необходимо* добавить «ro» в строку `kernel`, поскольку JFS «накатывает» свой журнал перед тем, как разрешить монтирование раздела на чтение-запись.

Листинг 3.3: Пример /etc/lilo.conf

```
boot=/dev/hda          # установка LILO в MBR
prompt                 # предоставление шанса выбора другого варианта
timeout=50             # ожидание пять секунд до загрузки варианта по умолчанию
default=gentoo         # по истечении времени загрузка варианта gentoo

# для тех, кто не использует genkernel
image=/boot/kernel-2.6.12-gentoo-r10
  label=gentoo         # название этого варианта
  read-only            # запуск с корневой ФС только для чтения; не менять!
  root=/dev/hda3       # расположение корневой файловой системы

# для тех, кто использует genkernel
image=/boot/kernel-genkernel-x86-2.6.12-gentoo-r10
  label=gentoo
  read-only
  root=/dev/ram0
  append="init=/linuxrc ramdisk=8192 real_root=/dev/hda3 udev"
  initrd=/boot/initramfs-genkernel-2.6.12-gentoo-r10

# следующие две строки нужны только для двойной загрузки с Windows
# в этом примере Windows находится на /dev/hda6
other=/dev/hda6
  label=windows
```

Примечание: Параметр `udev`, указанный в конце строки kernel, необходим для обхода ошибки в некоторых версиях genkernel, *если* вы вообще используете udev (по умолчанию — используется).

Примечание: Если вы разбили жесткий диск по-другому, или у вас другое ядро, внесите необходимые изменения.

Если нужно передать ядру дополнительные параметры, добавьте в соответствующий раздел файла выражение `append`. Например, добавим параметр `video` для включения кадрового буфера:

Листинг 3.4: Использование append для добавления параметров ядра

```
image=/boot/kernel-2.6.12-gentoo-r10
  label=gentoo
  read-only
  root=/dev/hda3
  append="video=vesafb:mtrr,ywrap,1024x768-32@85"
```

Если вы используете ядро 2.6.7 или выше, а объем жесткого диска ограничили перемычками из-за того, что BIOS не в состоянии работать с дисками большого размера, вам потребуется добавить `hdx=stroke`.

Тем, кто использует `genkernel`, нужно помнить, что их ядро использует такие же загрузочные параметры, как на установочном компакт-диске. Например, если у вас есть устройства SCSI, следует передать ядру параметр `doscsi`.

Теперь сохраните файл и выйдите из редактора. Для окончания установки нужно запустить `/sbin/lilo`, чтобы LILO смог отразить настройки, сделанные в `/etc/lilo.conf`, в вашей системе (т.е. записался на диск). Имейте в виду, что при каждой установке нового ядра или изменении меню вам потребуется выполнять `/sbin/lilo` заново.

Листинг 3.5: Завершение установки LILO

```
# /sbin/lilo
```

Примечание: При переустановке ядра вам больше не придется копировать файлы. Просто запустите `make install` после компиляции ядра; копирование файлов и изменение конфигурации LILO выполнится автоматически.

Переходите к перезагрузке системы.

10.d. Перезагрузка системы

Выйдите из изолированной среды и размонтируйте все смонтированные разделы. Затем введите ту самую волшебную команду, которую вы так заждались: `reboot`.

Листинг 4.1: Размонтирование всех разделов и перезагрузка

```
# exit
cdimage ~# cd
cdimage ~# umount /mnt/gentoo/boot /mnt/gentoo/dev /mnt/gentoo/proc /mnt/gentoo
cdimage ~# reboot
```

Естественно, не забудьте вынуть загрузочный компакт-диск, иначе он загрузится сам вместо вашей новой системы Gentoo.

Загрузив вновь установленную систему, переходите к завершению установки Gentoo.

11. Завершение установки Gentoo

11.a. Управление учетными записями

Добавление учетной записи для повседневной работы

Работа в учетной записи root (администратора) в системе Unix/Linux *опасна*, и ее следует всячески избегать. Поэтому *настоятельно* рекомендуется добавить учетную запись пользователя для повседневной работы.

Членством пользователя в группах определяется, какие действия он сможет выполнять. В следующей таблице перечислено несколько важных групп, в которые вы, возможно, захотите включать пользователей.

Группа	Описание
audio	возможность доступа к аудиоустройствам
cdrom	возможность прямого доступа к оптическим накопителям
floppy	возможность прямого доступа к гибким дискам
games	возможность играть в игры
portage	возможность использования `emerge --pretend` с правами пользователя
usb	возможность доступа к устройствам USB
plugdev	возможность монтирования и использования подключаемых устройств типа камер и USB-брелков
video	возможность доступа к средствам видеозахвата и выполнения аппаратного ускорения видео
wheel	возможность использования команды `su`

Например, для создания учетной записи пользователя по имени `john`, входящего в группы `wheel`, `users` и `audio`, сначала войдите в систему как root (только root может создавать учетные записи пользователей), а затем запустите `useradd`:

Листинг 1.1: Создание учетной записи на каждый день

```
Login: root
Password: (ваш пароль root)

# useradd -m -G users,wheel,audio -s /bin/bash john
# passwd john
Password: (введите пароль для john)
Re-enter password: (повторно введите пароль для подтверждения)
```

Если пользователю потребуется выполнить задачу от имени root, для временного получения привилегий root можно использовать `su -`. Другой способ — пользоваться пакетом `sudo`, который при правильной настройке вполне безопасен.

12. Чем заняться дальше?

12.a. Документация

Примите поздравления! У вас теперь появилась работающая система Gentoo. И что же делать дальше? Какие у вас появились возможности? На что стоит взглянуть прежде всего? Gentoo дает своим пользователям богатый выбор возможностей, а следовательно — и множество документированных (или не очень) свойств.

Вам обязательно нужно прочитать следующую часть настольной книги Gentoo, работа в Gentoo, в которой рассказывается, как поддерживать программное обеспечение в актуальном состоянии, как доустанавливать программы, что такое «USE-флаги», как происходит инициализация в Gentoo и т.д.

Если вас интересует оптимизация системы с точки зрения пользователя, или вы хотите узнать, как настроить полноценный «рабочий стол», обратитесь к более подробной документации по графической среде Gentoo. Кроме того, вы, возможно захотите прочитать наше руководство по локализации (англ.), что бы чуствовать себя более уютно.

Примечание: В настоящее время для русификации рекомендуется пользоваться альтернативными руководствами по локализации Gentoo. — *прим. пер.*

Также имеется настольная книга по безопасности в Gentoo (англ.), которую определенно стоит прочитать.

Полный список существующих материалов имеется на странице документации.

12.b. Gentoo в интернете

Естественно, мы всегда рады видеть вас на форумах Gentoo (англ.), как и на любом из множества IRC-

каналов Gentoo (англ.).

Кроме того, мы можем предложить вашему вниманию несколько списков рассылки, открытых для всех наших пользователей. Сведения о порядке подписки находятся на той же странице.

На этом мы замолкаем, чтобы позволить вам в полной мере насладиться результатом установки :)

B. Работа с Gentoo

1. Введение в Portage

1.a. Добро пожаловать в Portage

Система Portage — вероятно, самое известное нововведение Gentoo в управлении программным обеспечением. Благодаря высокой гибкости и чрезвычайно богатым возможностям, она зачастую считается лучшим средством управления программным обеспечением из существующих в Linux.

Portage полностью написана на Python и Bash, и в результате полностью прозрачна для пользователей, поскольку оба — языки сценариев.

Большинство пользователей взаимодействует с Portage с помощью команды `emerge`. Эта глава не призвана заменить страницу справки emerge. Для просмотра всех возможных параметров команды emerge, обращайтесь к странице справки:

Листинг 1.1: Чтение страницы справки emerge

```
$ man emerge
```

1.b. Дерево портежей

Сборочные файлы ebuild

Говоря о пакетах, мы часто имеем в виду программы, доступные пользователям Gentoo через дерево портежей. Дерево портежей — это набор *сборочных файлов ebuild*, содержащих всю информацию, необходимую Portage для управления программным обеспечением (установки, поиска, извлечения и т.п.) По умолчанию сборочные файлы находятся в `/usr/portage`.

Когда Portage по вашему поручению выполняет любые действия над пакетами программ, эти действия опираются на сборочные файлы, имеющиеся в системе. Поэтому необходимо регулярно обновлять сборочные файлы, чтобы Portage знала о новых программах, обновлениях, связанных с безопасностью и т.д.

Обновление дерева портежей

Дерево портежей обычно обновляется с помощью rsync (англ.), средства быстрой разностной передачи файлов. Обновление выполнить довольно просто, так как запуск rsync обеспечивается командой `emerge` :

Листинг 2.1: Обновление дерева портежей

```
# emerge --sync
```

Если rsync выполнить невозможно из-за ограничений межсетевого экрана, дерево портежей все-таки можно обновить из ежедневных «снимков», создаваемых нами. Для автоматического извлечения и установки в системе новейшего снимка служит утилита `emerge-webrsync`:

Листинг 2.2: Запуск emerge-webrsync

```
# emerge-webrsync
```

1.c. Обслуживание программного обеспечения

Поиск программ

Для поиска программ в дереве портежей по названию можно использовать встроенные возможности команды `emerge`. По умолчанию команда `emerge --search` выдает названия пакетов, соответствующих (как полностью, так и частично) заданному условию поиска.

Например, чтобы найти все пакеты, содержащие «pdf» в названии:

Листинг 3.1: Поиск пакетов с pdf в названии

```
$ emerge --search pdf
```

Для поиска пакетов еще и по тексту описания можно использовать параметр `--searchdesc` (или `-S`):

Листинг 3.2: Поиск пакетов, связанных с pdf

```
$ emerge --searchdesc pdf
```

Посмотрев на сообщения команды, вы отметите, что вам дается множество информации. Поля четко обозначены, поэтому мы не будем вдаваться в подробности их значения:

Листинг 3.3: Пример вывода emerge --search

```
*  net-print/cups-pdf
      Latest version available: 1.5.2
      Latest version installed: [ Not Installed ]
      Size of downloaded files: 15 kB
      Homepage:    http://cip.physik.uni-wuerzburg.de/~vrbehr/cups-pdf/
      Description: Provides a virtual printer for CUPS to produce PDF files.
      License:     GPL-2

(
*  net-print/cups-pdf
      Последняя доступная версия: 1.5.2
      Последняя установленная версия: [ не установлен ]
      Размер загружаемых файлов: 15 kB
      Веб-страница:  http://cip.physik.uni-wuerzburg.de/~vrbehr/cups-pdf/
      Описание:      Снабжает CUPS виртуальным принтером для печати PDF-файлов.
      Лицензия:      GPL-2 )
```

Установка программ

После того, как вы нашли нужное программное обеспечение, его можно легко установить с помощью команды `emerge`. Вот пример установки пакета `gnumeric`:

Листинг 3.4: Установка gnumeric

```
# emerge gnumeric
```

Так как множество приложений зависит друг от друга, любая попытка установить какой-либо пакет программ может повлечь за собой также установку дополнительных пакетов. Не беспокойтесь, Portage справится и с этим. Если вы захотите выяснить, что именно Portage собирается установить вместе с нужным вам пакетом, добавьте параметр `--pretend`. Например:

Листинг 3.5: Проверка зависимостей пакета gnumeric

```
# emerge --pretend gnumeric
```

После команды на установку пакета, Portage загружает из интернета необходимый исходный код (при необходимости), и по умолчанию сохраняет его в каталоге `/usr/portage/distfiles`. После этого пакет распаковывается, компилируется и устанавливается. Если вы хотите, чтобы Portage только загрузила

исходный код без его установки, добавьте к команде `emerge` параметр `--fetchonly`:

Листинг 3.6: Загрузка исходного кода пакета gnumeric

```
# emerge --fetchonly gnumeric
```

Обнаружение документации к пакету

Многие пакеты содержат собственную документацию. Иногда USE-флаг `doc` определяет, следует ли устанавливать документацию к пакету. Проверить наличие USE-флага `doc` можно командой `emerge -vp <название пакета>`.

Листинг 3.7: Проверка наличия USE-флага doc

```
(alsa-lib - это всего лишь пример)
# emerge -vp alsa-lib
[ebuild  N    ] media-libs/alsa-lib-1.0.9_rc3  +doc -jack 674 kB
```

USE-флаг `doc` можно включить или отключить как глобально в файле `/etc/make.conf`, так и для отдельных пакетов в файле `/etc/portage/package.use`. Также можно, создав каталог с названием `/etc/portage/package.use`, указать флаг в файле внутри этого каталога. В главе USE-флаги этот вопрос описывается более подробно.

Документация от вновь установленного пакета обычно находится в подкаталоге каталога `/usr/share/doc`, соответствующем названию пакета. Кроме того, можно вывести список всех установленных файлов утилитой `equery`, которая входит в пакет gentoolkit (англ.) — `app-portage/gentoolkit`.

Листинг 3.8: Обнаружение документации пакета

```
# ls -l /usr/share/doc/alsa-lib-1.0.9_rc3
total 28
-rw-r--r--  1 root root  669 May 17 21:54 ChangeLog.gz
-rw-r--r--  1 root root 9373 May 17 21:54 COPYING.gz
drwxr-xr-x  2 root root 8560 May 17 21:54 html
-rw-r--r--  1 root root  196 May 17 21:54 TODO.gz

(или используйте для поиска интересных файлов команду equery :)
# equery files alsa-lib | less
media-libs/alsa-lib-1.0.9_rc3
* Contents of media-libs/alsa-lib-1.0.9_rc3:
/usr
/usr/bin
/usr/bin/alsalisp
(выдача обрезана)
```

Удаление пакета

Когда вы захотите удалить пакет из системы, используйте команду `emerge --unmerge`. Это приведет к удалению из системы всех файлов, установленных пакетом, *кроме* конфигурационных файлов приложения, изменявшихся после установки. Сохранение конфигурационных файлов позволяет вернуться к работе с пакетом, если вы когда-нибудь решите снова его установить.

Внимание: Portage *не проверяет*, зависят ли другие пакеты от удаляемого! Однако вы получите предупреждение, если удаление пакета приведет к неработоспособности системы.

Листинг 3.9: Удаление пакета gnumeric из системы

```
# emerge --unmerge gnumeric
```

После удаления пакета из системы, пакеты, установленные автоматически, потому что от них зависел удаленный пакет, остаются. Чтобы Portage выявила все когда-то нужные пакеты, которые теперь можно удалить, используйте команду `emerge --depclean`. Мы вернемся к этому ниже.

Обновление системы

Чтобы система сохранялась в отличной форме (не говоря уже об установке свежайших обновлений, связанных с безопасностью), ее нужно регулярно обновлять. Так как Portage просматривает сборочные файлы только в локальном дереве портежей, сперва потребуется обновить его. Обновив дерево портежей, вы сможете обновить систему командой `emerge --update world`. В следующем примере мы также пользуемся параметром `--ask`, который поручает Portage вывести список пакетов, которые она собирается обновить, и спросить вас, можно ли продолжать:

Листинг 3.10: Обновление системы

```
# emerge --update --ask world
```

Portage будет искать более новые версии установленных приложений. Однако проверяется только версии приложений, явно установленных вами, а не тех, от которых они зависят. Если вы хотите обновить каждый пакет в системе, добавьте аргумент `--deep`:

Листинг 3.11: Обновление всей системы

```
# emerge --update --deep world
```

Поскольку обновления, относящиеся к безопасности, случаются и в пакетах, явным образом не устанавливались (но были «подтянуты» из-за того, что он них зависят другие программы), рекомендуется изредка запускать эту команду.

Если вы меняли какие-либо из USE-флагов, возможно, потом вы также захотите добавить параметр `--newuse`. Тогда Portage проверит, требует ли изменение установки новых пакетов или перекомпиляции существующих:

Листинг 3.12: Выполнение полного обновления

```
# emerge --update --deep --newuse world
```

Метапакеты

У некоторых пакетов в дереве портежей нет содержимого как такового, и они используются для установки набора других пакетов. Например, пакет `kde` полностью устанавливает среду KDE в вашей системе, привлекая различные KDE-пакетов в качестве зависимостей.

Если вы когда-либо захотите удалить из системы такой пакет, запуск `emerge--unmerge` не возымеет должного эффекта, так как пакеты, от которых он зависит, останутся в системе.

В Portage существует возможность удаления остаточных зависимосей, но так как зависимости программ меняются со временем, доступность программного обеспечения, прежде всего требуется полностью обновить всю систему, включая реализацию изменений, произведенных путем модификации USE-флагов. После этого можно запустить `emerge --depclean`, чтобы удалить остаточные зависимости. Когда это сделано, вам потребуется пересобрать приложения, ранее динамически связанные с удаленными пакетами, в которых они теперь не нуждаются.

Со всем этим управляются следующие три команды:

Листинг 3.13: Удаление ненужных зависимостей

```
# emerge --update --deep --newuse world
# emerge --depclean
# revdep-rebuild
```

`revdep-rebuild` входит в пакет `gentoolkit`; не забудьте сначала его установить:

Листинг 3.14: Установка пакета gentoolkit

```
# emerge gentoolkit
```

1.d. Когда Portage жалуется...

Слоты, виртуалы, ветви, архитектуры и профили

Как уже сказано, Portage — чрезвычайно мощная система, поддерживающая множество возможностей, не хватающих другим системам управления программами. Чтобы это стало понятно, разберем несколько аспектов Portage, не вникая в подробности.

С помощью Portage разные версии отдельного пакета могут сосуществовать в одной системе. В то время, как другие системы управления стремятся называть пакеты в соответствии с версией (например `freetype` и `freetype2`), в Portage используется технология *слотов* (SLOT), или областей. Пакет присваивает определенный слот своей версии. Пакеты с разными слотами способны сосуществовать в одной системе. Например, у пакета `freetype` есть ebuild как со `SLOT="1"`, так и со `SLOT="2"`.

Существуют также пакеты, выполняющие одни и те же функции, но отличающиеся в реализации. Например `metalogd`, `sysklogd` и `syslog-ng` являются системными службами журналирования. Приложения, использующие «системный журнал», не могут зависеть от одной конкретной программы, например от `metalogd`, так как остальные программы ничем не хуже. В Portage предусмотрены *виртуальные пакеты*: каждая служба журналирования предоставляет `virtual/syslog`, и в результате в приложениях можно указывать зависимость от `virtual/syslog`.

Программное обеспечение может располагаться в различных ветвях дерева портежей. По умолчанию в системе разрешено только использование стабильных пакетов. Большинство новых программ при поступлении включаются в тестовую ветвь, что указывает на необходимость дополнительного тестирования перед тем, как включить их в стабильные. Хотя в дереве портежей и видны сборочные файлы для таких программ, Portage не станет обновлять их до тех пор, пока они не будут помещены в стабильную ветвь.

Некоторые программы имеются не для всех архитектур. Либо они не работают в определенных архитектурах, либо требуют дополнительного тестирования, или у разработчика нет возможности проверить, работает ли пакет в различных архитектурах.

Каждая установка Gentoo придерживается определенного `профиля`, который содержит, помимо прочего, список пакетов, необходимых для работоспособности системы.

Блокировка пакетов

Листинг 4.1: Предупреждение о заблокированных пакетах (с --pretend)

```
[blocks B     ] mail-mta/ssmtp (is blocking mail-mta/postfix-2.2.2-r1)
```

Листинг 4.2: Предупреждение о заблокированных пакетах (без --pretend)

```
!!! Error: the mail-mta/postfix package conflicts with another package.
!!!        both can't be installed on the same system together.
!!!        Please use 'emerge --pretend' to determine blockers.

( !!! Ошибка: пакет mail-mta/postfix конфликтует с другим пакетом.
  !!! оба не могут находиться в системе одновременно. Пожалуйста,
  !!! запустите 'emerge --pretend' для выявления блокирующих пакетов. )
```

В файлах ebuild есть специальные поля, сообщающие Portage о зависимостях. Возможны два вида зависимости: зависимость сборки, объявленная в `DEPEND`, и зависимость выполнения, объявленная в `RDEPEND`. Когда одна из этих зависимостей явно указывает на *несовместимость* пакета или виртуального пакета, это вызывает блокировку.

Для разблокировки можно отказаться от установки пакета или предварительно удалить конфликтующего пакета. В данном примере можно отказаться от установки `postfix` или сначала удалить `ssmtp`.

Также возможно, что два пакета, подлежащие установке, блокируют друг друга. В этом редчайшем случае следует определить, зачем вам устанавливать оба пакета. В большинстве случаев можно обойтись одним. Если это окажется не так, то, пожалуйста, заведите отчет об ошибке в системе распределения запросов Gentoo.

Маскировка пакетов

Листинг 4.3: Предупреждение о замаскированных пакетах

```
!!! all ebuilds that could satisfy "bootsplash" have been masked.

(!!! все сборки, удовлетворяющие "bootsplash", замаскированы.)
```

Листинг 4.4: Предупреждение о замаскированных пакетах с указанием причины

```
!!! possible candidates are:

- gnome-base/gnome-2.8.0_pre1 (masked by: ~x86 keyword)
- lm-sensors/lm-sensors-2.8.7 (masked by: -sparc keyword)
- sys-libs/glibc-2.3.4.20040808 (masked by: -* keyword)
- dev-util/cvsd-1.0.2 (masked by: missing keyword)
- media-video/ati-gatos-4.3.0 (masked by: package.mask)
- sys-libs/glibc-2.3.2-r11 (masked by: profile)

( !!! возможные кандидаты:

- gnome-base/gnome-2.8.0_pre1 (маскировка: ключ ~x86)
- lm-sensors/lm-sensors-2.8.7 (маскировка: ключ -sparc)
- sys-libs/glibc-2.3.4.20040808 (маскировка: ключ -*)
- dev-util/cvsd-1.0.2 (маскировка: ключ отсутствует)
- media-video/ati-gatos-4.3.0 (маскировка: package.mask)
- sys-libs/glibc-2.3.2-r11 (маскировка: profile) )
```

Когда вы собираетесь установить пакет, не предназначенный для вашей системы, выдается ошибка маскировки. Нужно попытаться установить другую программу, существующую для вашей системы, или дождаться, пока пакет станет доступным. Всегда есть причина, по которой пакет замаскирован:

- **ключ ~arch**: пакет недостаточно проверен для помещения в стабильную ветвь. Подождите несколько дней или недель и попробуйте установить его еще раз.
- **ключ -arch** или **ключ -***: пакет не работоспособен в вашей архитектуре. Если вы полагаете, что он работает, сообщите об этом в bugzilla.
- **ключ отсутствует**: пакет еще не тестировался в вашей архитектуре. Попросите группу портирования в архитектуру проверить пакет, или протестируйте его за них и сообщите о своих изысканиях в bugzilla.
- **package.mask**: обнаружено повреждение пакета, нестабильность или что-то худшее, и пакет заблокирован специально.
- **profile**: пакет считается не предназначенным для вашего профиля. В случае установки приложение может вызвать сбой системы или просто несовместимо с используемым профилем.

Отсутствие нужных пакетов

Листинг 4.5: Предупреждение об отсутствии пакета

```
emerge: there are no ebuilds to satisfy ">=sys-devel/gcc-3.4.2-r4".

!!! Problem with ebuild sys-devel/gcc-3.4.2-r2
!!! Possibly a DEPEND/*DEPEND problem.

( emerge: нет сборок, удовлетворяющих ">=sys-devel/gcc-3.4.2-r4".

  !!! Проблема с ebuild sys-devel/gcc-3.4.2-r2
  !!! Возможно, ошибка в DEPEND/*DEPEND. )
```

Приложение, которое вы пытаетесь установить, зависит от другого пакета, недоступного вашей системе. Пожалуйста, проверьте, есть ли такой запрос в bugzilla, а если нет, сообщите об ошибке. Если вы не смешиваете ветви, такого не должно происходить, и это — явная ошибка.

Неоднозначность названия пакета

Листинг 4.6: Предупреждение о повторяющихся именах ebuild

```
!!! The short ebuild name "aterm" is ambiguous.  Please specify
!!! one of the following fully-qualified ebuild names instead:
```

```
    dev-libs/aterm
    x11-terms/aterm

( !!! Короткое название ebuild "aterm" неоднозначно. Пожалуйста,
  !!! вместо него укажите одно из полных названий ebuild:

      dev-libs/aterm
      x11-terms/aterm )
```

Название приложения, которое вы собираетесь установить, соответствует более чем одному пакету. Требуется также указать название категории. Portage предложит вам возможные варианты.

Циклические зависимости

Листинг 4.7: Предупреждение Portage о циклических зависимостях

```
!!! Error: circular dependencies:

ebuild / net-print/cups-1.1.15-r2 depends on ebuild /
app-text/ghostscript-7.05.3-r1
ebuild / app-text/ghostscript-7.05.3-r1 depends on ebuild /
net-print/cups-1.1.15-r2

( !!! Ошибка: циклические зависимости:

  ebuild / net-print/cups-1.1.15-r2 зависит от ebuild /
  app-text/ghostscript-7.05.3-r1
  ebuild / app-text/ghostscript-7.05.3-r1 зависит от ebuild /
  net-print/cups-1.1.15-r2   )
```

Два или более пакета, которые вы хотите установить, взаимно зависимы, и в результате их установка невозможна. Скорее всего, это ошибка в дереве портежей. Пожалуйста, выждав время, обновите дерево портежей, и попытайтесь снова. Вы можете также проверить, есть ли эта ошибка в bugzilla, и если нет, сообщить о ней.

Ошибка извлечения

Листинг 4.8: Предупреждение Portage об ошибке извлечения

```
!!! Fetch failed for sys-libs/ncurses-5.4-r5, continuing...
(...)
!!! Some fetch errors were encountered.  Please see above for details.

( !!! Ошибка при извлечении sys-libs/ncurses-5.4-r5, продолжение...
  (...)
!!! При извлечении произошли ошибки.  Подробности выше. )
```

Portage не смогла загрузить исходный код данного приложения и попытается продолжить установку других приложений (если запланирована). Эта ошибка может произойти из-за неправильно синхронизированного зеркала, или из-за того, что ebuild указывает на неверное место. Сервер, где находятся исходные коды, также может почему-либо не работать.

Повторите действие через час, чтобы посмотреть, повторится ли эта ошибка.

Защита системного профиля

Листинг 4.9: Предупреждение Portage о пакете, защищенном профилем

```
!!! Trying to unmerge package(s) in system profile. 'sys-apps/portage'
!!! This could be damaging to your system.

( !!! Попытка удаления пакетов из системного профиля. 'sys-apps/portage'
  !!! Это может повредить вашей системе. )
```

Вы попросили удалить пакет, входящий в состав базовых пакетов вашей системы. Он отмечен в вашем профиле как обязательный, и его не следует удалять из системы.

2. USE-флаги

2.a. Что такое USE-флаги?

Смысл USE-флагов

Устанавливая Gentoo (или любой другой дистрибутив, или даже операционную систему вообще), вы выбираете те или иные возможности в зависимости от среды, с которой работаете. Установка сервера отличается от установки рабочей станции, а установка игровой станции — от платформы 3D-рендеринга.

Это касается не только того, какие пакеты устанавливать, но и какие функции определенных пакетов должны поддерживаться. Если вам не нужен OpenGL, то зачем вам его ставить и встраивать поддержку OpenGL в большинство программ? Если вы не собираетесь использовать KDE, зачем собирать пакеты с его поддержкой, если они работают и без этого?

Чтобы помочь пользователям в выборе того, что устанавливать/активировать, а что — нет, мы захотели дать им простой способ описания рабочей среды. Это позволяет пользователю решить, что же ему на самом деле надо, и облегчить работу с Portage, нашей системой управления пакетами.

Определение USE-флагов

Рассмотрим USE-флаги. USE-флаг — это ключевое слово, включающее сведения о поддержке и зависимостях определенного понятия или функции. При определении какого-либо USE-флага, Portage узнает, что вам нужна поддержка соответствующей функции. Конечно, это также влияет на сведения о зависимостях пакета.

Давайте рассмотрим конкретный пример — ключевое слово `kde`. Если в вашей переменной `USE` нет этого слова, то все пакеты, где поддержка KDE является *необязательной*, собираются *без* нее. Все пакеты, где зависимость от KDE является *необязательной*, устанавливаются *без* установки библиотек KDE (по зависимости). Если же вы определите ключевое слово `kde`, то эти пакеты будут собираться *с поддержкой* KDE, а KDE будет установлен в качестве необходимого.

Правильно определяя ключевые слова, вы создаете систему, подогнанную специально для ваших нужд.

Какие USE-флаги существуют?

Есть два типа USE-флагов: *глобальные* и *локальные*.

- *Глобальный* USE-флаг используется несколькими пакетами и является системным. Это то, что большинство видит в качестве USE-флагов.
- *Локальный* USE-флаг используется единичным пакетом для настройки определенных параметров самого пакета.

Список доступных глобальных USE-флагов можно найти в сети или локально в `/usr/portage/profiles/use.desc`.

Список локальных USE-флагов находится в вашей системе в `/usr/portage/profiles/use.local.desc`.

2.b. Использование USE-флагов

Объявление постоянных USE-флагов

В надежде, что вы убедились в важности USE-флагов, теперь мы расскажем, как их объявлять.

Как сказано ранее, все USE-флаги объявляются в переменной `USE`. Чтобы упростить пользователям поиск и выбор флагов, мы предлагаем значение USE *по умолчанию*, которое представляют собой список USE-флагов, как нам кажется, наиболее часто используемых пользователями Gentoo. Это значение приведено в файле `make.defaults` вашего профиля.

Профиль, на который ориентируется ваша система, указывается символьной ссылкой `/etc/make.profile`. Каждый профиль основывается на предыдущем, более крупном, а итоговый складывается из всех профилей. Верхним является `базовый` профиль (`/usr/portage/profiles/base`).

Давайте взглянем на значение по умолчанию для профиля 2004.3:

Листинг 2.1: Итоговая переменная USE make.defaults для профиля 2004.3

```
(в этом примере объединяются значения из base, default-linux,
default-linux/x86 и default-linux/x86/2004.3)
USE="x86 oss apm arts avi berkdb bitmap-fonts crypt cups encode fortran f77
     foomaticdb gdbm gif gpm gtk imlib jpeg kde gnome libg++ libwww mad
     mikmod motif mpeg ncurses nls oggvorbis opengl pam pdflib png python qt
     quicktime readline sdl spell ssl svga tcpd truetype X xml2 xmms xv zlib"
```

Как видите, эта переменная уже содержит достаточно много ключевых слов. **Не меняйте** файл `make.defaults` для настройки переменной `USE` под свои нужды: изменения в этом файле аннулируются при обновлении Portage!

Для изменения значения по умолчанию, нужно добавлять или удалять ключевые слова из переменной `USE` . Это делается глобально, определением переменной `USE` в `/etc/make.conf`. В эту переменную можно добавить нужные вам USE-флаги, или удалить ненужные. Для удаления флага, его надо указывать со знаком минус в виде приставки («-»).

Например, чтобы убрать поддержку KDE и QT, но добавить поддержку ldap, можно определить в `/etc/make.conf` переменную `USE` следующего вида:

Листинг 2.2: Пример значения USE в /etc/make.conf

```
USE="-kde -qt ldap"
```

Объявление USE-флагов для отдельных пакетов

Иногда нужно определить некоторые USE-флаги только для одного или нескольких пакетов, не трогая системных настроек. Для этого необходимо создать каталог `/etc/portage` (если его еще нет) и отредактировать файл `/etc/portage/package.use`.

Например, вам не нужна глобальная поддержка `berkdb`, но она необходима в `mysql`:

Листинг 2.3: Пример /etc/portage/package.use

```
dev-db/mysql berkdb
```

Естественно, можно в явном виде `отключить` USE-флаги для определенного пакета. Например, если вам не нужна поддержка `java` в PHP:

Листинг 2.4: Второй пример /etc/portage/package.use

```
dev-php/php -java
```

Объявление временных USE-флагов

Иногда необходимо установить какой-то USE-флаг только на один раз. Вместо того, чтобы дважды редактировать `/etc/make.conf` (сначала добавить изменения USE, а потом удалить), можно просто объявить USE как переменную среды. Помните, что при переустановке или обновлении приложения (явном или в составе обновления системы) ваши изменения будут утеряны!

Например, уберем java из значения USE на время установки mozilla.

Листинг 2.5: Использование USE в виде переменной среды

```
# USE="-java" emerge mozilla
```

Наследование

Конечно же, существует определенная последовательность формирования значения USE. Вы же не хотите объявить `USE="-java"` только для того, чтобы узнать, что `java` все еще включена из-за значения с более высоким приоритетом. Последовательность установки значения USE в порядке приоритета (от меньшего к большему) такова:

1. значение USE по умолчанию, объявленное в файлах `make.defaults` в составе вашего профиля
2. значение, определенное пользователем в `/etc/make.conf`
3. значение, указанное пользователем в `/etc/portage/package.use`
4. значение, определенное пользователям в переменной среды

Чтобы узнать, какие же настройки `USE` в конечном счете видит Portage, запустите `emerge --info`. Эта команда выводит значения всех переменных (включая `USE`), используемые Portage.

Листинг 2.6: Запуск emerge --info

```
# emerge --info
```

Адаптация всей системы под новые USE-флаги

Если вы изменили свои USE-флаги и хотите обновить всю систему в соответствии с новым значением USE, запустите `emerge` с параметром `--newuse`:

Листинг 2.7: Пересборка всей системы

```
# emerge --update -deep --newuse world
```

Теперь запустите функцию Portage depclean, чтобы удалить условные зависимости, присутствующие в «старой» системе, но больше не нужные при новом составе USE-флагов.

Предупреждение: Запуск `emerge --depclean` является опасной операцией, которую следует использовать с осторожностью. Дважды проверьте список «ненужных» пакетов и убедитесь, что не удалятся нужные пакеты. В следующем примере мы добавляем ключ `-p`, чтобы depclean только перечислил пакеты, не удаляя их.

Листинг 2.8: Удаление ненужных пакетов

```
# emerge -p --depclean
```

Когда depclean закончит свою работу, запустите `revdep-rebuild`, чтобы пересобрать программы, динамически связанные с библиотеками, входящими в потенциально удаленные пакеты. `revdep-rebuild` входит в пакет `gentoolkit`, так что не забудьте сначала установить его.

Листинг 2.9: Запуск revdep-rebuild

```
# revdep-rebuild
```

После выполнения всех этих действий, ваша система будет полностью использовать новые значения USE-флагов.

2.c. USE-флаги отдельных пакетов

Просмотр доступных USE-флагов

Возьмем, к примеру, `mozilla` — какие USE-флаги она может использовать? Чтобы это выяснить, запустим `emerge` с параметрами `--pretend` и `--verbose`:

Листинг 3.1: Просмотр используемых USE флагов

```
# emerge --pretend --verbose mozilla
```

```
These are the packages that I would merge, in order:

Calculating dependencies ...done!
[ebuild   R   ] www-client/mozilla-1.7.12-r2  USE="crypt gnome java mozsvg ssl
truetype xprint -debug -ipv6 -ldap -mozcalendar -mozdevelop -moznocompose
-moznoirc -moznomail -moznoxft -postgres -xinerama" 0 kB
```

emerge — не единственное средство для решения этой задачи. Существует программа, специально предназначенная для вывода информация о пакетах. Она называется equery и находится в пакете gentoolkit. Для начала установим этот пакет:

Листинг 3.2: Установка gentoolkit

```
# emerge gentoolkit
```

Теперь для просмотра USE-флагов какого-нибудь пакета запустим equery с аргументом uses. Пусть это будет пакет gnumeric:

Листинг 3.3: Запуск equery для просмотра доступных USE-флагов

```
# equery uses =gnumeric-1.6.3 -a
[ Searching for packages matching =gnumeric-1.6.3... ]
[ Colour Code : set unset ]
[ Legend        : Left column  (U) - USE flags from make.conf  ]
[               : Right column (I) - USE flags packages was installed with ]
[ Found these USE variables for app-office/gnumeric-1.6.3 ]
 U I
- - debug   : Tells configure and the makefiles to build for debugging.
                    Effects vary across packages, but generally it will at
                    least add -g to CFLAGS.  Remember to set FEATURES=nostrip too
- - gnome   : Adds GNOME support
+ + python   : Adds support/bindings for the Python language
- - static   : !!do not set this during bootstrap!! Causes binaries to be
                    statically linked instead of dynamically
```

3. Возможности Portage

3.a. Возможности Portage

В Portage есть несколько дополнительных возможностей (features), которые значительно улучшат ваше впечатление от Gentoo. Многие из этих возможностей полагаются на определенные программы, повышающие производительность, надежность, безопасность и т.п.

Для включения и выключения определенных возможностей Portage нужно редактировать в файле /etc/make.conf переменную FEATURES, в которой перечислены ключевые слова, разделенные пробелами, обозначающие различные возможности. Иногда для использования соответствующих возможностей потребуется установка дополнительных утилит.

Здесь перечислены не все возможности, поддерживаемые Portage. Полный перечень представлен на странице справки make.conf:

Листинг 1.1: Вызов страницы справки make.conf

```
$ man make.conf
```

Чтобы узнать, какие возможности включены по умолчанию, запустите emerge --info и найдите переменную FEATURES (или отфильтруйте ее с помощью grep):

Листинг 1.2: Выявление уже включенных возможностей

```
$ emerge --info | grep FEATURES
```

3.b. Распределенная компиляция

Использование distcc

`distcc` — программа, распределяющая компиляцию по нескольким, не обязательно одинаковым, машинам в сети. Клиент `distcc` посылает всю необходимую информацию на доступные серверы distcc (на которых выполняется `distccd`), чтобы они могли компилировать для клиента части исходного кода. Чистый выигрыш — более быстрая компиляция.

Подробная информация о `distcc` (и как заставить его заработать в Gentoo) находится в нашем описании distcc в Gentoo.

Установка distcc

Distcc поставляется с графическим монитором (средством контроля), позволяющим отслеживать задачи, которые ваш компьютер отсылает на компиляцию. Если вы используете Gnome, тогда добавьте «gnome» к переменной USE. А если вы не пользуетесь Gnome, но при этом хотите пользоваться монитором, добавьте «gtk» к переменной USE.

Листинг 2.1: Установка distcc

```
# emerge distcc
```

Подключение поддержки Portage

Добавьте `distcc` к переменной FEATURES в файле `/etc/make.conf`. Затем отредактируйте переменную MAKEOPTS, как вам нравится. Известная рекомендация — указывать директиву «-jX», где X — число центральных процессоров, на которых работает `distccd` (включая текущий компьютер) плюс один; у вас могут получиться лучшие результаты и с другими значениями.

Теперь запустите `distcc-config` и введите список доступных серверов distcc. Для простоты примера, предположим, что доступные серверы DistCC — 192.168.1.102 (текущий компьютер), 192.168.1.103 и 192.168.1.104 (два «удаленных» компьютера):

Листинг 2.2: Настройка distcc для использования трех доступных серверов distcc

```
# distcc-config --set-hosts "192.168.1.102 192.168.1.103 192.168.1.104"
```

Не забудьте также запустить демон `distccd`:

Листинг 2.3: Запуск демонов distccd

```
# rc-update add distccd default
# /etc/init.d/distccd start
```

3.c. Кэширование компиляции

О средстве ccache

`ccache` — это быстрый кэш компилятора. Когда вы компилируете программу, он кэширует промежуточные результаты так, что всякий раз, когда вы перекомпилируете ту же самую программу, время компиляции значительно сокращается. В типичных случаях общее время компиляции может сокращаться в 5—10 раз.

Если вы интересуетесь подробностями ccache, пожалуйста, посетите домашнюю страницу ccache.

Установка ccache

Для установки `ccache`, выполните `emerge ccache`:

Листинг 3.1: Установка ccache

```
# emerge ccache
```

Подключение поддержки Portage

Откройте `/etc/make.conf` и добавьте `ccache` к переменной FEATURES. Затем добавьте новую переменную по имени CCACHE_SIZE (размер кэша), и установите её равной «2G»:

Листинг 3.2: Редактирование CCACHE_SIZE в /etc/make.conf

```
CCACHE_SIZE="2G"
```

Для проверки работоспособности ccache, запросите статистику ccache. Из-за того, что Portage использует другой домашний каталог ccache, вам также потребуется установить переменную `CCACHE_DIR`:

Листинг 3.3: Просмотр статистики ccache

```
# CCACHE_DIR="/var/tmp/ccache" ccache -s
```

Домашний каталог ccache по умолчанию — `/var/tmp/ccache`; изменить это назначение можно, определив переменную `CCACHE_DIR` в `/etc/make.conf`.

Однако, при запуске `ccache` используется каталог по умолчанию, `${HOME}/.ccache`, вот почему при запросе статистики (Portage) ccache требуется определять переменную `CCACHE_DIR`.

Использование ccache для компиляции Си не в Portage

Если вы хотите использовать ccache для компиляций не в Portage, добавьте `/usr/lib/ccache/bin` в начало вашей переменной PATH (перед `/usr/bin`). Это можно сделать, отредактировав `/etc/env.d/00basic`, который является первым файлом среды, где определяется переменная `PATH`:

Листинг 3.4: Редактирование /etc/env.d/00basic

```
PATH="/usr/lib/ccache/bin:/opt/bin"
```

3.d. Поддержка двоичных пакетов

Создание готовых (заранее собранных) пакетов

Portage поддерживает установку заранее собранных готовых пакетов. Несмотря на то, что в саму Gentoo не входят заранее собранные пакеты (за исключением снимков GRP), Portage можно настроить на полноценное управление готовыми пакетами.

Чтобы создать двоичный пакет, можно использовать `quickpkg`, если пакет уже установлен в вашей системе, или `emerge` с параметрами `--buildpkg` или `--buildpkgonly`.

Если вы хотите, чтобы Portage создавал двоичные пакеты из каждого пакета, который вы будете устанавливать, добавьте `buildpkg` к переменной FEATURES.

Расширенная поддержка создания наборов готовых пакетов имеются в `catalyst`. Для получения подробной информации о catalyst, пожалуйста, прочитайте справочное руководство по catalyst (англ.) и распространенные вопросы о catalyst (англ.).

Установка двоичных пакетов

Хотя в Gentoo такого хранилища нет, вы можете создать централизованное хранилище для заранее скомпилированных двоичных пакетов. Чтобы использовать такое хранилище, потребуется указать Portage путь к нему с помощью переменной PORTAGE_BINHOST. Например, если двоичные пакеты находятся на ftp://buildhost/gentoo:

Листинг 4.1: Установка PORTAGE_BINHOST в /etc/make.conf

```
PORTAGE_BINHOST="ftp://buildhost/gentoo"
```

При установке двоичных пакетов, указывайте в команде emerge параметр `--getbinpkg` вместе с параметром `--usepkg`. Первый указывает emerge загрузить двоичный пакет с сервера, определенного раньше, а второй сообщает emerge, что до загрузки исходных кодов и их компиляции сначала нужно попытаться установить этот двоичный пакет.

Например, чтобы установить `gnumeric` из двоичных пакетов:

Листинг 4.2: Установка двоичного пакета gnumeric

```
# emerge --usepkg --getbinpkg gnumeric
```

Подробную информацию о параметрах установки двоичных пакетов можно найти на странице справки emerge:

Листинг 4.3: Чтение справки по emerge

```
$ man emerge
```

4. Сценарии инициализации

4.a. Уровни запуска

Процесс загрузки системы

При загрузке вашей системы по экрану пробегает много текста. Если присмотреться, заметно, что этот текст не меняется от загрузки к загрузке. Последовательность всех этих действий называется *последовательностью загрузки* и в той или иной степени постоянна.

Во-первых, загрузчик размещает в памяти образ ядра, который вы указали в файле его конфигурации. После этого ядро запускается. Когда ядро загружено и запущено, оно инициализирует относящиеся к ядру структуры и задания, и запускает процесс `init`.

Этот процесс удостоверяется, что все файловые системы (определенные в `/etc/fstab`) смонтированы и готовы к использованию. Затем он выполняет несколько сценариев, находящихся в каталоге `/etc/init.d`, которые запускают службы, необходимые для нормального запуска системы.

И, наконец, когда все сценарии выполнены, `init` подключает терминалы (чаще всего просто виртуальные консоли, которые видны при нажатии `ALT+F1`, `ALT+F2` и т.д.), прикрепляя к каждой консоли специальный процесс под названием `agetty`. Этот процесс впоследствии обеспечивает возможность входа в систему с помощью `login`.

Сценарии инициализации

Сейчас процесс `init` запускает сценарии из каталога `/etc/init.d` не просто в случайном порядке. Более того, запускаются не все сценарии из `/etc/init.d`, а только те, которые предписано исполнять. Решение о запуске сценария принимается в результате просмотра каталога `/etc/runlevels`.

Во-первых, `init` запускает все сценарии из `/etc/init.d`, на которые есть символьные ссылки из `/etc/runlevels/boot`. Обычно сценарии запускаются в алфавитном порядке, но в некоторых сценариях имеется информация о зависимостях от других сценариев, указывающая системе на необходимость их предварительного запуска.

Когда все сценарии, указанные в `/etc/runlevels/boot`, будут выполнены, `init` переходит к запуску сценариев, на которые есть символьные ссылки из `/etc/runlevels/default`. И снова запуск происходит в алфавитном порядке, пока в сценарии не встретится информация о зависимостях; тогда порядок изменяется для обеспечения правильного порядка запуска.

Как работает init

Конечно, `init` не принимает решений сам по себе. Ему необходим конфигурационный файл, где описаны необходимые действия. Этот файл — `/etc/inittab`.

Если вы запомнили последовательность загрузки, описанную чуть ранее, вы вспомните, что первое действие `init` — это монтирование всех файловых систем. Это определяется в строке `/etc/inittab`, приведенной ниже:

Листинг 1.1: Строка инициализации системы из /etc/inittab

```
si::sysinit:/sbin/rc sysinit
```

Этой строкой процессу `init` предписывается выполнить `/sbin/rc sysinit` для инициализации системы. Самой инициализацией занимается сценарий `/sbin/rc`, так что можно сказать, что `init` делает не слишком много — он просто делегирует задачу по инициализации системы другому процессу.

Во-вторых, `init` выполняет все сценарии, на которые есть символьные ссылки из `/etc/runlevels/boot`. Это определяется следующей строкой:

Листинг 1.2: Инициализация системы, продолжение

```
rc::bootwait:/sbin/rc boot
```

И снова все необходимые действия выполняются сценарием `rc`. Заметьте, что параметр, переданный `rc` (*boot*), совпадает с названием используемого подкаталога в `/etc/runlevels`.

Теперь `init` проверяет свой конфигурационный файл, чтобы определить, какой *уровень запуска* использовать. Для этого из `/etc/inittab` считывается строка:

Листинг 1.3: Строка initdefault

```
id:3:initdefault:
```

В приведенном примере (который подходит для подавляющего большинства пользователей Gentoo) номер *уровня запуска* — 3. Пользуясь этой информацией, `init` проверяет, что нужно выполнить для запуска *уровня запуска 3*:

Листинг 1.4: Определение уровней запуска

```
l0:0:wait:/sbin/rc shutdown
l1:S1:wait:/sbin/rc single
l2:2:wait:/sbin/rc nonetwork
l3:3:wait:/sbin/rc default
l4:4:wait:/sbin/rc default
l5:5:wait:/sbin/rc default
l6:6:wait:/sbin/rc reboot
```

В строке, определяющей уровень 3, для запуска служб снова используется сценарий `rc` (на этот раз с аргументом *default*). Опять-таки, обратите внимание, что аргумент, передаваемый сценарию `rc`, совпадает с названием подкаталога из `/etc/runlevels`.

По окончании работы `rc`, `init` принимает решение о том, какие виртуальные консоли включить и какие команды выполнить в каждой из них:

Листинг 1.5: Определение виртуальных консолей

```
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:12345:respawn:/sbin/agetty 38400 tty2 linux
c3:12345:respawn:/sbin/agetty 38400 tty3 linux
c4:12345:respawn:/sbin/agetty 38400 tty4 linux
c5:12345:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux
```

Что такое уровень запуска?

Как вы заметили, `init` применяет нумерацию для определения *уровня запуска*, который надо использовать. *Уровень запуска* — это то состояние, в котором запускается ваша система, он содержит набор сценариев (сценариев уровня запуска или *сценариев инициализации [initscript]*), которые следует выполнять, при входе и выходе из определенного уровня запуска.

В Gentoo определено семь уровней запуска: три служебных и четыре определяемых пользователем. Служебные называются *sysinit*, *shutdown* и *reboot*. Действия, совершаемые ими, в точности соответствуют их названиям: инициализация системы, выключение системы и ее перезагрузка.

Определяемые пользователем уровни — это те, которым соответствуют подкаталоги в `/etc/runlevels`: `boot`, `default`, `nonetwork` и `single`. Уровень `boot` запускает все службы, необходимые системе и используемые всеми остальными уровнями. Остальные уровни отличаются друг от друга запускаемыми службами: `default` используется для повседневной работы, `nonetwork` — для тех случаев, когда не требуется сеть, а `single` — при необходимости восстановления системы.

Работа со сценариями инициализации

Сценарии, запускаемые процессом `rc`, называются *сценариями инициализации*. Каждый сценарий из `/etc/init.d` может запускаться с аргументами *start*, *stop*, *restart*, *pause*, *zap*, *status*, *ineed*, *iuse*, *needsme*, *usesme* и *broken*.

Для запуска, остановки или перезапуска службы (и всех, зависящих от нее) следует использовать `start`, `stop` и `restart`:

Листинг 1.6: Запуск postfix

```
# /etc/init.d/postfix start
```

Примечание: Останавливаются или перезапускаются только те службы, которым *необходима* данная служба. Остальные зависимые службы (те, которые *используют* службу, но не нуждаются в ней) эта операция не затрагивает.

Если вы хотите остановить службу, но оставить зависимые от нее работающими, можно использовать аргумент `pause`:

Листинг 1.7: Остановка postfix без остановки зависимых служб

```
# /etc/init.d/postfix pause
```

Чтобы узнать текущее состояние службы (запущена, остановлена, приостановлена и т.д.), можно использовать аргумент `status`:

Листинг 1.8: Информация о состоянии postfix

```
# /etc/init.d/postfix status
```

Если указано, что служба работает, но вы знаете, что это не так, можно сбросить состояние на stopped (остановлена), используя аргумент `zap`:

Листинг 1.9: Сброс информации о состоянии postfix

```
# /etc/init.d/postfix zap
```

Для того, чтобы выяснить зависимости службы, можно использовать аргументы `iuse` или `ineed`. С помощью `ineed` вы увидите те службы, которые действительно необходимы для правильного функционирования интересующей вас службы. С другой стороны, `iuse` покажет те службы, которые могут использоваться нашей службой, но не обязательны для ее работы.

Листинг 1.10: Запрос списка всех необходимых служб, от которых зависит Postfix

```
# /etc/init.d/postfix ineed
```

Аналогично вы можете узнать, какие службы нуждаются в данной службе (`needsme`) или могут ее использовать (`usesme`):

Листинг 1.11: Запрос списка всех служб, которым необходим Postfix

```
# /etc/init.d/postfix needsme
```

Наконец, можно просмотреть список служб, требующихся для данной, но отсутствующих в системе:

Листинг 1.12: Запрос списка служб, необходимых Postfix, но отсутствующих

```
# /etc/init.d/postfix broken
```

4.b. Использование rc-update

Что такое rc-update?

Система инициализации Gentoo использует дерево зависимостей для определения служб, которые запускаются в первую очередь. Т. к. это очень утомительное занятие, и мы не хотели, чтобы пользователь занимался этим вручную, мы разработали инструменты, упрощающие управление уровнями запуска и сценариями инициализации.

Используя `rc-update`, можно включать и исключать сценарии инициализации из уровней запуска. Из `rc-update` автоматически запускается сценарий `depscan.sh` для перестроения дерева зависимостей.

Добавление и удаление служб

В процессе установки Gentoo вы уже добавляли сценарии инициализации в уровень запуска «default». В тот момент вы, возможно, не имели понятия, что такое «default» и зачем он нужен, но теперь вы все это знаете. Сценарию `rc-update` требуется второй аргумент, определяющий действие: *add* (добавить), *del* (удалить) или *show* (показать).

Для того, чтобы добавить или удалить сценарий, просто введите `rc-update` с аргументом `add` или `del`, затем название сценария и уровня запуска. Например:

Листинг 2.1: Удаление Postfix из уровня запуска default

```
# rc-update del postfix default
```

По команде `rc-update show` выводится список всех доступных сценариев с указанием соответствующих уровней запуска:

Листинг 2.2: Получение информации о сценариях инициализации

```
# rc-update show
```

4.c. Настройка служб

Почему нужна дополнительная настройка?

Сценарии инициализации могут быть весьма сложны. Поэтому нежелательно допускать непосредственное редактирование сценария пользователями, т.к. это может привнести в систему множество ошибок. Но, с другой стороны, необходимо правильно настроить службу. Например, может понадобиться передать службе дополнительные параметры.

Вторая причина, по которой настройки хранятся отдельно от самого сценария — это возможность обновления сценария без опасения, что все ваши настройки будут утеряны.

Каталог /etc/conf.d

В Gentoo предусмотрен очень простой способ настройки служб: для каждого сценария, предполагающего настройку, в каталоге `/etc/conf.d` есть конфигурационный файл. Например, у сценария, запускающего apache2 (под названием `/etc/init.d/apache2`) есть конфигурационный файл `/etc/conf.d/apache2`, где могут храниться нужные вам параметры, передаваемые серверу Apache 2 при запуске:

Листинг 3.1: Переменная, определенная в /etc/conf.d/apache2

```
APACHE2_OPTS="-D PHP4"
```

Такие файлы настроек содержат одни переменные (наподобие `/etc/make.conf`), облегчая настройку служб. Это также позволяет нам давать больше информации о переменных (в комментариях).

4.d. Написание сценариев инициализации

Мне тоже придется?..

Нет, написание сценариев инициализации обычно не требуется, т.к. Gentoo содержит готовые сценарии для всех поддерживаемых служб. Однако, вы можете установить какую-либо службу, не используя систему Portage; в таком случае, вероятно, вам придется создавать сценарий инициализации самостоятельно.

Не используйте сценарий, идущий со службой, если он не написан специально для Gentoo: сценарии инициализации Gentoo не совместимы со сценариями, используемыми в других дистрибутивах!

Структура

Основная структура сценария инициализации показана ниже.

Листинг 4.1: Основная структура сценария

```
#!/sbin/runscript

depend() {
   (информация о зависимостях)
}

start() {
   (команды, необходимые для запуска службы)
}

stop() {
   (команды, необходимые для остановки службы)
}

restart() {
   (команды, необходимые для перезапуска службы)
}
```

В любом сценарии *должна* быть определена функция `start()`. Все остальные разделы необязательны.

Зависимости

Можно определять два типа зависимостей: `use` (использую) и `need` (нуждаюсь). Как упоминалось ранее, `need`-зависимость более строга, чем `use`-зависимость. Вслед за типом зависимости указывается название службы, от которой существует зависимость, или ссылка на *виртуальную* (virtual) зависимость.

Виртуальная зависимость — это зависимость от функций, предоставляемых службой, но не какой-то единственной службой. Сценарий может зависеть от службы системного журнала, но таких достаточно много (metalogd, syslog-ng, sysklogd и т.п.). Поскольку нельзя нуждаться в каждой из них (ни в одной вразумительной системе они не запущены все сразу), мы обеспечили `предоставление` виртуальной зависимости всеми этими службами.

Давайте взглянем на информацию о зависимостях postfix.

Листинг 4.2: Информация о зависимостях Postfix

```
depend() {
  need net
  use logger dns
  provide mta
}
```

Как можно увидеть, postfix:

- требует сеть (`net`): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/net.eth0`
- использует журнал (`logger`): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/syslog-ng`
- использует службу имен (`dns`): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/named`)
- предоставляет почтовый агент (`mta`): виртуальная зависимость, общая для всех программ — почтовых серверов

Порядок запуска

Иногда вам нужна не сама служба, а запуск вашей службы `до` (или *после*) другой службы, *если* та присутствует в системе (обратите внимание на условие: это уже не зависимость) *и* запускается на том же уровне запуска (отметьте условие: это относится только к службам из одинакового уровня запуска). Такую очередность можно указать, используя значения `before` (до) или `after` (после).

Например, рассмотрим значения для службы Portmap:

Листинг 4.3: Функция depend() службы Portmap

```
depend() {
  need net
  before inetd
  before xinetd
}
```

Также можно использовать знак «*«, чтобы охватить все службы данного уровня запуска, хотя это не рекомендуется.

Листинг 4.4: Запуск сценария первым на уровне запуска

```
depend() {
  before *
}
```

Стандартные функции

Следом за разделом `depend()` вам потребуется определить функцию `start()`. В ней содержатся все команды, необходимые для запуска вашей службы. Рекомендуется применять функции `ebegin` и `eend` для сообщений пользователю о том, что происходит:

Листинг 4.5: Пример функции start()

```
start() {
  ebegin "Запуск - моя_служба"
  start-stop-daemon --start --quiet --exec /path/to/my_service
  eend $?
}
```

Если вам нужны дополнительные примеры функции `start()`, пожалуйста, прочитайте исходные коды

сценариев инициализации, находящихся в каталоге `/etc/init.d`. Что касается команды `start-stop-daemon`, то на случай, если вам нужны дополнительные сведения, есть превосходная страница справки:

Листинг 4.6: Вызов страницы справки по start-stop-daemon

```
# man start-stop-daemon
```

Другими функциями, которые можно определить — `stop()` и `restart()`. От вас не требуется определение этих функций! Система инициализации, применяемая нами, достаточно развита и в состоянии самостоятельно заполнить эти функции, если вы используете `start-stop-daemon`.

Синтаксис сценариев инициализации, применяемых в Gentoo, основан на оболочке Борна (Bourne Again Shell — bash), поэтому вы можете свободно использовать внутри своих сценариев bash-совместимые конструкции.

Добавление дополнительных параметров

Если вы хотите ввести в сценарий дополнительные параметры, кроме упоминавшихся, нужно добавить к переменной `opts` название параметра и создать функцию с названием, соответствующим параметру. Например, для поддержки параметра `restartdelay`:

Листинг 4.7: Создание дополнительной функции restartdelay

```
opts="${opts} restartdelay"

restartdelay() {
  stop
  sleep 3    # пауза в 3 секунды перед повторным запуском
  start
}
```

Переменные для настройки служб

Для поддержки конфигурационного файла в каталоге `/etc/conf.d` ничего дополнительно делать не нужно: при запуске вашего сценария инициализации автоматически включаются следующие файлы (т.е., переменные из них становятся доступны):

- `/etc/conf.d/<ваш сценарий инициализации>`
- `/etc/conf.d/basic`
- `/etc/rc.conf`

Если ваш инициализационный сценарий предоставляет виртуальную зависимость (например, `net`), то также включается файл, соответствующий этой зависимости (например, `/etc/conf.d/net`).

4.e. Изменение поведения уровней запуска

Кто от этого выиграет?

Большинству пользователей ноутбуков знакома ситуация: дома вам нужен запуск `net.eth0`, и наоборот, в дороге запуск `net.eth0` не нужен (так как сеть недоступна). В Gentoo можно изменять поведение уровней запуска по своему усмотрению.

Например вы можете создать второй загружаемый уровень запуска «по умолчанию», в котором будут другие сценарии. Затем при загрузке вы сможете выбрать, какой из уровней по умолчанию следует использовать.

Использование программного уровня (softlevel)

Прежде всего, создайте каталог для своего второго уровня запуска «по умолчанию». Например, создадим уровень запуска `offline`:

Листинг 5.1: Создание каталога уровня запуска

```
# mkdir /etc/runlevels/offline
```

Добавьте необходимые сценарии инициализации в только что созданный уровень запуска. Например, чтобы получить точную копию уровня `default`, за исключением `net.eth0`:

Листинг 5.2: Добавление нужных сценариев инициализации

```
(копирование всех служб с уровня default в уровень offline)
# cd /etc/runlevels/default
# for service in *; do rc-update add $service offline; done
(удаление ненужных сценариев с уровня offline)
# rc-update del net.eth0 offline
(просмотр сценариев, запускаемых на уровне offline)
# rc-update show offline
(часть выведенного списка)
               acpid | offline
          domainname | offline
               local | offline
            net.eth0 |
```

Теперь необходимо отредактировать конфигурацию загрузчика, добавив запись об уровне `offline`. Например, в файле `/boot/grub/grub.conf`:

Листинг 5.3: Добавление записи об уровне offline

```
title Автономное использование Gentoo Linux
  root (hd0,0)
  kernel (hd0,0)/kernel-2.4.25 root=/dev/hda3 softlevel=offline
```

Вуаля, все готово. Теперь, если при загрузке вы выберете вновь созданную запись, то вместо `default` будет использоваться уровень `offline`.

Использование загрузочного уровня (bootlevel)

Использование `загрузочного уровня` полностью аналогично использованию `программного уровня`. Единственная разница состоит в том, что вы определяете второй уровень «boot» вместо «default».

5. Переменные среды

5.a. Переменные среды

Что это такое?

Переменная среды — это именованный объект, который содержит информацию, используемую одним или несколькими приложениями. Многие пользователи (особенно новички в Linux) находят этот подход несколько странным или неуправляемым. Но это впечатление ошибочно: используя переменные среды, можно очень легко изменить настройку разнообразных программ.

Важные примеры

В следующей таблице описывается ряд переменных, используемых в системе Linux. Примеры их значений приведены далее.

Переменная	Описание
PATH	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых система ищет исполняемые файлы. Если вы вводите имя исполняемого файла например, `ls`, `rc-update` или `emerge`), который не находится ни в одной из перечисленных здесь каталогов, этот файл не запустится (если, конечно, вы не указали полный путь, например, `/bin/ls`).**
ROOTPATH	**У этой переменной такое же значение, что и у `PATH`, но в ней перечисляются только те каталоги, которые нужно просматривать при вводе команды пользователем с правами root.**

LDPATH	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых динамический компоновщик ищет библиотеки.**
MANPATH	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых команда `man` ищет страницы справки.**
INFODIR	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых команда `info` ищет info-страницы.**
PAGER	**В этой переменной содержится путь к программе, позволяющей постранично просматривать содержимое файлов, например `less` или `more`.**
EDITOR	**В этой переменной содержится путь к программе, используемой для изменения файлов, например `vi` или `nano`.**
KDEDIRS	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых находятся ресурсы KDE.**
CLASSPATH	**В этой переменной содержится список каталогов, разделенных двоеточиями, в которых находятся классы Java.**
CONFIG_PROTECT	**В этой переменной содержится список каталогов, защищаемых Portage при обновлении, разделенных `пробелами`.**
CONFIG_PROTECT_MASK	**В этой переменной содержится список каталогов, исключаемых из защиты Portage при обновлении, разделенных `пробелами`**

Ниже представлен пример определения всех этих переменных:

Листинг 1.1: Пример определения

```
PATH="/bin:/usr/bin:/usr/local/bin:/opt/bin:/usr/games/bin"
ROOTPATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin"
LDPATH="/lib:/usr/lib:/usr/local/lib:/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
MANPATH="/usr/share/man:/usr/local/share/man"
INFODIR="/usr/share/info:/usr/local/share/info"
PAGER="/usr/bin/less"
EDITOR="/usr/bin/vim"
KDEDIRS="/usr"
CLASSPATH="/opt/blackdown-jre-1.4.1/lib/rt.jar:."
CONFIG_PROTECT="/usr/X11R6/lib/X11/xkb /opt/tomcat/conf \
                /usr/kde/3.1/share/config /usr/share/texmf/tex/generic/config/ \
                /usr/share/texmf/tex/platex/config/ /usr/share/config"
CONFIG_PROTECT_MASK="/etc/gconf"
```

5.b. Глобальное определение переменных

Каталог /etc/env.d

Для того, чтобы определить эти переменные централизованно, в Gentoo появился каталог `/etc/env.d`. В нём находится ряд файлов, например, `00basic`, `05gcc` и так далее, в которых определяются переменные, необходимые программам, указанным в названии файлов.

Например, при установке `gcc` ebuild создает файл `05gcc`, содержащий следующие определения переменных:

Листинг 2.1: /etc/env.d/05gcc

```
PATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
ROOTPATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
MANPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/man"
INFOPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/info"
CC="gcc"
CXX="g++"
LDPATH="/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
```

В других дистрибутивах вам предлагается изменять или добавлять определения переменных среды в `/etc/profile` или где-нибудь еще. Gentoo, с другой стороны, облегчает вам (и Portage) поддержку и управление переменными среды, избавляя от необходимости уделять внимание многочисленным файлам, содержащим определения переменных.

Например, когда обновляется `gcc`, также без малейшего участия пользователя обновляется и `/etc/env.d/05gcc`.

От этого выигрывает не только Portage, но и вы, пользователь. Иногда от вас может потребоваться глобальная установка какой-нибудь переменной. Возьмем, к примеру, переменную `http_proxy`. Вместо того, чтобы возиться с `/etc/profile`, теперь можно просто создать файл (`/etc/env.d/99local`) и добавить нужные определения туда:

Листинг 2.2: /etc/env.d/99local

```
http_proxy="proxy.server.com:8080"
```

Используя один и тот же файл для всех своих переменных, вы можете быстро увидеть все определенные вами переменные вместе.

Сценарий env-update

Переменная `PATH` определяется в нескольких файлах в `/etc/env.d`. Нет, нет это не ошибка: при запуске `env-update` различные определения объединяются перед обновлением переменных среды, позволяя пакетам (или пользователям) добавлять собственные значения переменных, не влияя на уже существующие.

Сценарий `env-update` объединяет значения переменных из файлов, находящихся в `/etc/env.d`, в алфавитном порядке. Имена файлов должны начинаться с двух десятичных цифр.

Листинг 2.3: Порядок обновления, используемый env-update

```
        00basic         99kde-env       99local
    +-------------+----------------+-------------+
PATH="/bin:/usr/bin:/usr/kde/3.2/bin:/usr/local/bin"
```

Объединение выполняется не всегда, а только для следующих переменных: `KDEDIRS`, `PATH`, `CLASSPATH`, `LDPATH`, `MANPATH`, `INFODIR`, `INFOPATH`, `ROOTPATH`, `CONFIG_PROTECT`, `CONFIG_PROTECT_MASK`, `PRELINK_PATH` и `PRELINK_PATH_MASK`. Для всех остальных переменных используется значение, определенное в последнем из файлов (по алфавиту в каталоге `/etc/env.d`).

При запуске сценария `env-update` создаются все переменные среды, и помещаются в `/etc/profile.env` (используемый файлом `/etc/profile`). Кроме того, на основе значения `LDPATH` создается `/etc/ld.so.conf`. После этого запускается `ldconfig`, чтобы вновь создать файла `/etc/ld.so.cache`, используемый динамическим компоновщиком.

Если вы хотите, чтобы результаты работы `env-update` вступили в силу немедлено, для обновления среды выполните следующую команду. Пользователи, самостоятельно устанавливавшие Gentoo, возможно, помнят ее из указаний по установке:

Листинг 2.4: Обновление среды

```
# env-update && source /etc/profile
```

Примечание: Эта команда обновляет переменные только в текущем терминале, в *новых* консолях и их потомках. То есть, если вы работаете в X11, потребуется или набирать `source /etc/profile` в каждом открываемом терминале, или перезапустить X, чтобы все новые терминалы обращались к новым переменным. Если вы используете диспетчер входа в систему, станьте пользователем с правами root и наберите `/etc/init.d/xdm restart`. Если нет, вам придется выйти и снова войти в систему, чтобы X порождала потомков, использующих новые значения переменных.

5.c. Локальное определение переменных

Пользовательские переменные

Далеко не всегда нужно определять переменные глобально. Например, вам может понадобиться добавить `/home/my_user/bin` и текущий рабочий каталог (где вы находитесь) к переменной `PATH`, но при этом не нужно, чтобы это добавление появилось и в переменной `PATH` у всех остальных пользователей. Если вы хотите определить переменную среды локально, используйте `~/.bashrc` или `~/.bash_profile`:

Листинг 3.1: Расширение PATH в ~/.bashrc для локальных нужд

```
(двоеточие без последующего указания каталога означает текущий рабочий каталог)
PATH="${PATH}:/home/my_user/bin:"
```

Обновление вашей переменной `PATH` произойдет, когда вы выйдете и снова войдете в систему.

Сеансовые переменные

Иногда нужны еще более жесткие ограничения. Вам может потребоваться возможность запуска исполняемых файлов из специально созданного временного каталога без указания полного пути к ним, и без изменения файла `~/.bashrc` ради нескольких минут.

В этом случае можно просто определить переменную `PATH` для текущего сеанса командой `export`. Переменной будет присвоено временное значение до тех пор, пока вы не завершите сеанс.

Листинг 3.2: Определение сеансовой переменной среды

```
# export PATH="${PATH}:/home/my_user/tmp/usr/bin"
```

C. Работа с Portage

1. Файлы и каталоги

1.a. Файлы Portage

Директивы настройки

Настройки Portage по умолчанию хранятся в `/etc/make.globals`. Когда вы откроете этот файл, вы увидите, что все настройки представляют собой переменные. Что означает каждая из переменных, описано ниже.

Так как многие директивы отличаются в зависимости от используемой архитектуры, к Portage прилагаются настройки по умолчанию, которые входят в ваш профиль. На ваш профиль указывает символическая ссылка `/etc/make.profile`. Настройка Portage выполняется с помощью файлов `make.defaults` вашего профиля и всех родительских профилей. Более подробно о профилях и каталоге `/etc/make.profile` мы расскажем позже.

Если вы планируете вносить изменения в конфигурационные переменные, *не* изменяйте `/etc/make.globals` или `make.defaults`. Вместо этого пользуйтесь файлом `/etc/make.conf`, который имеет приоритет перед вышеуказанными файлами. Вы также обнаружите файл `/etc/make.conf.example`. Как понятно из его названия, это просто пример — Portage не использует этот файл.

Переменные Portage также можно устанавливать как переменные среды, но мы не рекомендуем этого делать.

Конфигурация, определяемая профилем

Мы уже встречались с каталогом `/etc/make.profile`. На самом деле это не каталог, а символическая ссылка на профиль, по умолчанию на тот, что содержится в `/usr/portage/profiles`, однако вы можете создавать свои собственные профили где угодно и ссылаться на них. Профиль, указанный ссылкой, является профилем, к которому принадлежит ваша система.

В профиле содержатся сведения для Portage, специфичные для архитектуры, такие как список пакетов, принадлежащих соответствующей системе, список неработоспособных (или замаскированных) пакетов, и т.д.

Конфигурация, задаваемая пользователем

Если вам необходимо изменить поведение Portage относительно установки программного обеспечения, вам потребуется отредактировать файлы, находящиеся в `/etc/portage`. Мы *настоятельно рекомендуем* вам пользоваться файлами из `/etc/portage`, *всеми силами отговариваем* от настройки поведения Portage через переменные среды!

Внутри `/etc/portage` вы можете создать следующие файлы:

- `package.mask`, в котором перечислены пакеты, которые Portage никогда не следует устанавливать
- `package.unmask`, со списком пакетов, для которых вы хотите иметь возможность установки, даже если разработчики Gentoo отговаривают вас от этого
- `package.keywords`, где перечислены пакеты, которые должны быть доступны для установки, несмотря на то, что они не подходят для вашей системы или архитектуры (пока)
- `package.use`, где перечислены значения USE-флагов, которые необходимо указывать для конкретных пакетов, а не для всей системы

Дополнительные сведения о каталоге `/etc/portage`, а также список всех файлов, которые там можно создавать, находятся на справочной странице Portage:

Листинг 1.1: Вызов справки по Portage

```
$ man portage
```

Изменение файлов Portage и размещения каталогов

Ранее упомянутые конфигурационные файлы нельзя хранить где угодно — Portage всегда ищет свои настроечные файлы в строго определенных местах. Однако Portage также использует множество каталогов для других целей: каталог для сборки, место для хранения исходных кодов, место для дерева Portage, и т.д.

Для этих целей существуют хорошо известные каталоги по умолчанию, положение которых можно изменить на свой вкус, внеся изменения в `/etc/make.conf`. Оставшаяся часть этой главы посвящена описанию того, какие специальные места Portage использует для своих целей, и как изменить их расположение в файловой системе.

Этот документ не претендует на статус справочника. Если вам необходим полный объем информации, пожалуйста, обратитесь к страницам справки по Portage и `make.conf`:

Листинг 1.2: Вызов справки по Portage и make.conf

```
$ man portage
$ man make.conf
```

1.b. Хранение файлов

Дерево Portage

Дерево Portage размещается, по умолчанию, в `/usr/portage`. Это определяется значением переменной PORTDIR. Когда вы храните дерево Portage где-либо в другом месте (изменив эту переменную), не забывайте соответственно изменить символическую ссылку `/etc/make.profile`.

Если вы измените переменную PORTDIR, вам может потребоваться изменить и следующие переменные: PKGDIR, DISTDIR, RPMDIR, так как они не замечают изменений PORTDIR. Это связано с особенностями их обработки Portage.

Двоичные пакеты

Несмотря на то, что Portage по умолчанию не использует прекомпилированное программное обеспечение, для него предусмотрена очень мощная поддержка. Если вы укажете Portage работать с прекомпилированными пакетами, они будут разыскиваться в `/usr/portage/packages`. Это расположение определяется переменной PKGDIR.

Исходные коды

Исходные коды приложений хранятся в `/usr/portage/distfiles` по умолчанию. Это определяется переменной DISTDIR.

Файлы RPM

Несмотря на то, что Portage не может использовать RPM-файлы, есть возможность их создания командой `ebuild` (см. Приложение Ebuild). По умолчанию Portage хранит RPM файлы в каталоге `/usr/portage/rpm`, как определяется переменной RPMDIR.

База данных Portage

Portage хранит состояние вашей системы (какие пакеты установлены, какие файлы относятся к определенным пакетам и т. п.) в `/var/db/pkg`. *Не изменяйте* эти файлы вручную! Это может разрушить знание вашей системы Portage.

Кэш Portage

Кэш Portage (включая сведения о времени изменения, виртуальные пакеты, информацию дерева зависимостей и т. д.) хранится в `/var/cache/edb`. Это место действительно является кэшем: вы можете его очистить в любой момент, когда не запущены приложения, связанные с Portage.

1.c. Сборка программного обеспечения

Временные файлы Portage

По умолчанию Portage хранит временные файлы в `/var/tmp`. За это отвечает переменная PORTAGE_TMPDIR.

Если вы измените переменную PORTAGE_TMPDIR, вам может потребоваться изменить и переменную BUILD_PREFIX, так как она не замечает изменений PORTAGE_TMPDIR. Это связано с особенностями ее обработки Portage.

Каталог сборки

Portage создает специфичные каталоги сборки для каждого пакета внутри `/var/tmp/portage`. Это расположение задается переменной BUILD_PREFIX.

Размещение «живой файловой системы»

По умолчанию Portage устанавливает все файлы в текущую файловую систему (`/`), но это можно изменить, установив переменную окружения ROOT. Это может оказаться полезным при построении новых образов системы.

1.d. Ведение журнала

Журнал Ebuild

Portage может создавать отдельные файлы журнала для каждого файла ebuild, но только тогда, когда переменная PORT_LOGDIR указывает на место, доступное для записи для Portage (пользователя *portage*). По умолчанию эта переменная не установлена.

2. Настройка с помощью переменных

2.a. Настройка Portage

Как отмечалось ранее, Portage настраивается с помощью множества переменных, которые задаются в

файле `/etc/make.conf`. За более полной и подробной информацией обращайтесь к странице справки по `make.conf`:

Листинг 1.1: Чтение страницы справки по make.conf

```
$ man make.conf
```

2.b. Параметры сборки

Параметры конфигурирования и компиляции

Когда Portage собирает приложения, компилятору и сценарию конфигурации передаются значения следующих переменных:

- CFLAGS и CXXFLAGS определяют желаемые флаги компилятора для C и C++
- CHOST определяет информацию об используемой платформе для сценария конфигурации приложения
- MAKEOPTS передается команде `make` и обычно применяется для установки степени распараллеливания компиляции. Более подробная информация о параметрах команды make находится на странице справки по make.

Переменная USE также используется при конфигурировании и компиляции, но о ней уже много и подробно говорилось в предыдущих главах.

Параметры установки

Когда Portage устанавливает (merge) новую версию программного продукта, файлы более старых версий удаляются из системы. Portage дает пользователю 5-ти секундную задержку перед стиранием старых версий. Эти 5 секунд задаются переменной CLEAN_DELAY.

2.c. Защита конфигурационных файлов

Места, защищаемые Portage

Portage записывает файлы, предоставляемые новой версией программы, поверх старых, если только эти файлы не расположены в *защищенном* месте. Защищенные каталоги определяются переменной CONFIG_PROTECT. Обычно, это места расположения файлов конфигурации. Каталоги в списке разделяются пробелами.

Файл, который должен быть записан в такой защищенный каталог, переименовывается, а пользователь получает предупреждение о наличии новой версии (обычно) файла конфигурации.

Узнать текущее значение CONFIG_PROTECT можно из сообщений `emerge --info`:

Листинг 3.1: Получение значения CONFIG_PROTECT

```
$ emerge --info | grep 'CONFIG_PROTECT='
```

Более подробная информация о защите конфигурационных файлов, осуществляемой системой Portage, доступна по команде `emerge`:

Листинг 3.2: Подробная информация о защите конфигурационных файлов

```
$ emerge --help config
```

Исключение каталогов

Чтобы снять защиту с определенных подкаталогов защищенного каталога, можно использовать переменную CONFIG_PROTECT_MASK.

2.d. Параметры скачивания

Расположение сервера

Если запрошенная информация или данные отсутствуют в вашей системе, Portage обращается за ними в интернет. Расположение серверов для различных каналов получения информации задается следующими переменными:

- GENTOO_MIRRORS определяет список адресов серверов, содержащих исходный код (distfiles)
- PORTAGE_BINHOST указывает расположение определенного сервера, содержащего двоичные пакеты (prebuilt packages) для вашей системы

Третья переменная содержит расположение сервера rsync, который используется при обновлении вашего дерева портежей:

- SYNC указывает сервер, с которого Portage извлекает дерево портежей

Переменные GENTOO_MIRRORS и SYNC можно установить автоматически программой `mirrorselect`. Перед тем, как использовать, ее нужно установить, выполнив `emerge mirrorselect`. За дополнительной информацией обращайтесь к оперативной справке mirrorselect:

Листинг 4.1: Дополнительные сведения о mirrorselect

```
# mirrorselect --help
```

Если вы вынуждены использовать прокси-сервер, для его указания можно использовать переменные HTTP_PROXY, FTP_PROXY и RSYNC_PROXY.

Команды для извлечения

Когда Portage требуется извлечь исходный код, по умолчанию используется `wget`. Вы можете это изменить с помощью переменной FETCHCOMMAND.

Portage может возобновлять скачивание частично загруженного исходного кода. По умолчанию используется `wget`, но это можно переопределить переменной RESUMECOMMAND.

Удостоверьтесь, что ваши команды FETCHCOMMAND и RESUMECOMMAND сохраняют исходный код в нужном месте. Внутри этих переменных следует использовать \${URI} и \${DISTDIR}, для указания расположения исходных кодов и distfiles, соответственно.

Также существует возможность определить индивидуальные настройки для различных протоколов, используя FETCHCOMMAND_HTTP, FETCHCOMMAND_FTP, RESUMECOMMAND_HTTP, RESUMECOMMAND_FTP, и т.п.

Настройки rsync

Вы не можете заменить команду rsync, которую Portage использует для обновления дерева портежей, но можно установить несколько переменных, определяющих ее поведение:

- RSYNC_EXCLUDEFROM указывает на файл, где перечислены пакеты и/или категории, которые rsync должна игнорировать во время обновления.
- RSYNC_RETRIES определяет, сколько раз rsync должна пытаться соединиться с зеркалом, на которое указывает переменная SYNC. По умолчанию равна 3.
- RSYNC_TIMEOUT определяет количество секунд, в течение которого rsync соединение может бездействовать, перед тем как rsync сочтет его превысившим время ожидания. По умолчанию равна 180, но если вы используете соединение по модему или у вас медленный компьютер, возможно, следует установить значение этой переменной равным 300 или большим.

2.e. Настройка Gentoo

Выбор ветви

Используемую ветвь можно изменить переменной ACCEPT_KEYWORDS. По умолчанию используется стабильная ветвь для вашей архитектуры. Дополнительная информация о ветвях Gento находится в следующей главе.

Возможности Portage

Вы можете включить отдельные функции Portage с помощью переменной FEATURES. Возможности Portage рассматривались в предыдущих главах, например, Возможности Portage.

2.f. Поведение Portage

Распределение ресурсов

С помощью переменной PORTAGE_NICENESS можно увеличивать или уменьшать значение nice, с которым выполняется Portage. Значение PORTAGE_NICENESS *прибавляется* к текущему значению nice.

Более подробно о значениях nice написано в странице справки:

Листинг 6.1: Дополнительные сведения о nice

```
$ man nice
```

Настройки вывода

Переменная NOCOLOR (по умолчанию «false») определяет, следует ли Portage отключить цветовую раскраску своих сообщений.

3. Смешение ветвей программного обеспечения

3.a. Использование одной ветви

Стабильная ветвь

Переменная ACCEPT_KEYWORDS определяет, какую из ветвей использовать в вашей системе. По умолчанию используется стабильная ветвь для вашей архитектуры, например `x86`

Мы рекомендуем использовать только стабильную ветвь. Однако, если для вас стабильность не критична и вы хотите помочь Gentoo, отсылая отчеты об ошибках на http://bugs.gentoo.org, читайте дальше.

Тестовая ветвь

Если вы желаете использовать наиболее свежее ПО, подумайте над использованием тестовой ветви. Чтобы Portage начала использовать тестовую ветвь, добавьте «~» перед названием вашей архитектуры.

Тестовая ветвь полностью соответствует своему названию: *для тестирования*. Если пакет находится в стадии тестирования, это означает, что разработчики считают, что пакет работоспособен, но тщательно он не протестирован. Вы можете оказаться первым, кто столкнется с какой-либо ошибкой. В этом случае вы можете создать отчет об ошибке, чтобы разработчики узнали о ней.

Однако будьте готовы к тому, что могут возникнуть проблемы со стабильностью, неудовлетворительной поддержкой пакетов (например неправильные/отсутствующие зависимости), слишком частыми обновлениями (а в результате — частыми сборками) или невозможностью собрать пакет. Если вы не знаете, как работает Gentoo и как разрешать возникающие проблемы, мы рекомендуем не отходить от стабильной и оттестированной ветви.

К примеру, для выбора тестовой ветви на архитектуре x86, отредактируйте `/etc/make.conf` и укажите в нем:

Листинг 1.1: Установка значения переменной ACCEPT_KEYWORDS

```
ACCEPT_KEYWORDS="~x86"
```

Если вы запустите обновление системы, то увидите, что *многие* пакеты нуждаются в обновлении. Обратите внимание, что после перехода на тестовую ветвь и обновления системы, как правило, нет простого пути назад к стабильной официальной ветви (конечно, кроме использования резервной копии).

3.b. Одновременное использование стабильной и тестовой ветвей

Местоположение package.keywords

Вы можете указать, чтобы Portage использовала тестовую ветвь только для определенных пакетов, а для остальной системы — стабильную ветвь. Для этого добавьте категорию и имя пакета, для которого вы желаете использовать тестовую ветвь, в файл `/etc/portage/package.keywords`. Вместо этого можно создать каталог (с таким же именем) и указывать пакеты в файлах, находящихся внутри этого каталога. Например, для использования тестовой ветви для `gnumeric`:

Листинг 2.1: Настройка /etc/portage/package.keywords для gnumeric, вся строка

```
app-office/gnumeric ~x86
```

Тестирование определенных версий

Если вы желаете использовать конкретную версию ПО из тестовой ветви, но не хотите, чтобы Portage использовала тестовую ветвь для последующих версий этого ПО, можно указать в местоположении `package.keywords` номер необходимой версии. В этом случае вы *обязаны* использовать оператор =. Также можно указать диапазон версий, используя операторы <=, <, > или >=.

В любом случае, добавляя информацию о версии, вы *должны* использовать один из этих операторов. Если вы не указываете версию, эти операторы использовать *нельзя*.

В следующем примере мы просим Portage разрешить установку gnumeric-1.2.13:

Листинг 2.2: Использование конкретной тестовой версии gnumeric

```
=app-office/gnumeric-1.2.13 ~x86
```

3.c. Использование заблокированных пакетов

Расположение package.unmask

Разработчики Gentoo **не** поддерживают использование этого места расположения. Пожалуйста, используйте их на свой страх и риск. Просьбы о помощи, связанные с использованием `package.unmask` и/или `package.mask`, останутся без ответа. Вы предупреждены.

Если использование пакета было заблокировано разработчиками Gentoo, но вы желаете его использовать несмотря на причины блокировки, указанные в файле `package.mask` (по умолчанию он находится в `/usr/portage/profiles`), добавьте для него *точно такую же* строку в файл `/etc/portage/package.unmask` (или в файл в этом каталоге, если это каталог).

Например, если `=net-mail/hotwayd-0.8` заблокирован, то разблокировать его можно, прописав в `package.unmask` точно такую же строчку:

Листинг 3.1: /etc/portage/package.unmask

```
=net-mail/hotwayd-0.8
```

Местоположение package.mask

Если вы не хотите, чтобы Portage использовала какое-то конкретное ПО или конкретные версии ПО, вы можете его самостоятельно заблокировать, добавив соответствующую запись в `/etc/portage/package.mask` (в такой файл либо в файл внутри такого каталога).

Если, к примеру, вы не хотите, чтобы Portage устанавливала исходные коды ядра новее, чем `gentoo-sources-2.6.8.1`, добавьте такую строку в местоположение `package.mask`:

Листинг 3.2: Пример использования файла /etc/portage/package.mask

```
>sys-kernel/gentoo-sources-2.6.8.1
```

4. Дополнительные средства Portage

4.a. etc-update

`etc-update` — это утилита, предназначенная для обновления в системе файлов `._cfg0000_<имя>`. Она обеспечивает интерактивную настройку установки и может также автоматически устанавливать тривиальные изменения. Файлы создаются `._cfg0000_<имя>` Portage, когда нужно заменить файл в каталоге, защищенном переменной CONFIG_PROTECT.

Выполнить `etc-update` довольно просто:

Листинг 1.1: Запуск etc-update

```
# etc-update
```

После выполнения тривиальных обновлений, вы увидите запрос со списком защищенных файлов, ожидающих обновления. Внизу вам предложат следующие варианты:

Листинг 1.2: Запрос etc-update

```
Please select a file to edit by entering the corresponding number.
              (-1 to exit)  (-3 to auto merge all remaining files)
                            (-5 to auto-merge AND not use 'mv -i'):

(Пожалуйста, выберите файл для правки, введя соответствующее число.
          (-1 - выход)  (-3 - автоустановка всех оставшихся файлов)
                   (-5 для автоустановки БЕЗ использования 'mv -i'): )
```

При вводе `-1`, `etc-update` выходит, прекращая последующие изменения. Если вы введете `-3` или `-5`, *все* перечисленные файлы конфигурации заменяются более новыми версиями. Следовательно, очень важно сначала отобрать файлы, которые не следует автоматически обновлять. Для этого надо только вводить номер, указанный слева от файлов.

Например, выбираем файл конфигурации `/etc/pear.conf`:

Листинг 1.3: Обновление конкретного конфигурационного файла

```
Beginning of differences between /etc/pear.conf and /etc/._cfg0000_pear.conf
[...]
End of differences between /etc/pear.conf and /etc/._cfg0000_pear.conf
1) Replace original with update
2) Delete update, keeping original as is
3) Interactively merge original with update
4) Show differences again
```

Теперь можно увидеть различия между двумя файлами. Если вы считаете, что обновленный файл конфигурации можно использовать без проблем, введите `1`. Если вы считаете, что обновленный файл конфигурации не нужен, или не содержит новую или полезную информацию, введите `2`. Если вы хотите обновить текущий файл в интерактивном режиме, введите `3`.

Нет никакого смысла в подробном описании интерактивного обновления. Для полноты изложения, мы

перечислим возможные команды, которые можно использовать при интерактивном слиянии двух файлов. Вас встречают две строки (одна исходная, вторая измененная) и запрос, в ответ на который можно ввести одну из следующих команд:

Листинг 1.4: Команды, доступные при интерактивном слиянии

```
ed:     редактировать и использовать оба варианта, каждый пометить заголовком
eb:     редактировать и использовать оба варианта
el:     редактировать и использовать левый вариант
er:     редактировать и использовать правый вариант
e:      редактировать новую версию
l:      использовать левую версию
r:      использовать правую версию
s:      молча включить общие строки
v:      включить общие строки, сообщив подробности
q:      выход
```

Завершив обновление важных файлов конфигурации, вы можете автоматически обновить оставшиеся файлы конфигурации. `etc-update` выйдет, если не найдет других файлов, подлежащих обновлению.

4.b. dispatch-conf

С помощью `dispatch-conf` можно обновлять файлы конфигурации, сохраняя при этом историю изменений. `dispatch-conf` хранит различия между файлами конфигурации в виде заплаток или в системе управления версиями RCS.

Как и с `etc-update`, вы можете попросить сохранить файл конфигурации как есть, использовать новый файл конфигурации, редактировать текущий или объединить изменения интерактивно. Однако, у `dispatch-conf` также есть приятные дополнительные возможности:

- автоматическое обновление файлов, в которых обновились только комментарии
- автоматическое обновление файлов, которые отличаются только количеством пробелов

Убедитесь, что вы сначала отредактировали `/etc/dispatch-conf.conf` и создали каталог, прописанный в archive-dir.

За дополнительными сведениями обращайтесь к странице справки `dispatch-conf`:

Листинг 2.1: Чтение справки по dispatch-conf

```
$ man dispatch-conf
```

4.c. quickpkg

С `quickpkg` вы можете создавать архивы пакетов, уже установленных в системе. Эти архивы можно использовать в качестве двоичных пакетов. Запуск `quickpkg` прост: только укажите имена пакетов, которые нужно заархивировать.

Например, чтобы поместить в архив `curl`, `arts` и `procps`:

Листинг 3.1: Пример использования quickpkg

```
# quickpkg curl arts procps
```

Двоичные пакеты будут храниться в `$PKGDIR/All` (по умолчанию — `/usr/portage/packages/All`). Символьные ссылки, указывающие на эти пакеты, помещаются в `$PKGDIR/<категория>`.

5. Отступление от официального дерева

5.a. Использование собственного дерева Portage

Исключение пакета/категории

Вы можете выборочно обновлять определенные категории/пакеты, игнорируя обновление других категорий/пакетов. Это достигается путем исключения таких категорий/пакетов программой `rsync` на этапе выполнения `emerge --sync`.

Вам потребуется определить имя файла, содержащего шаблоны исключаемых пакетов, в переменной `RSYNC_EXCLUDEFROM` в своем файле `/etc/make.conf`.

Листинг 1.1: Указание файла исключаемых пакетов в /etc/make.conf

```
RSYNC_EXCLUDEFROM=/etc/portage/rsync_excludes
```

Листинг 1.2: Исключение всех игр в файле /etc/portage/rsync_excludes

```
games-*/*
```

Заметьте, однако, что это может привести к проблемам с зависимостями, так как новые разрешенные пакеты могут зависеть от других новых, но исключенных из обновления пакетов.

5.b. Добавление неофициального сборочного файла ebuild

Определение оверлейного каталога портежей

Вы можете указать Portage использовать сборочные файлы, не входящие в официальное дерево Portage. Создайте новый каталог (к примеру, `/usr/local/portage`), в котором будут находиться файлы ebuild сторонних разработчиков. Используйте в точности такую же структуру каталогов, как и в официальном дереве портежей!

Затем определите переменную PORTDIR_OVERLAY в `/etc/make.conf`, чтобы она указывала на ранее созданный каталог. Теперь при использовании Portage, эти сборочные файлы будут рассматриваться как часть системы, и не будут удаляться/перезаписываться при последующих запусках `emerge --sync`.

Работа с несколькими оверлейными каталогами

Для продвинутых пользователей, ведущих разработку в нескольких оверлейных каталогах, тестирующих пакеты перед включением в основное дерево портежей или просто желающих использовать неофициальные сборочные файлы ebuild из разных источников, в пакете `app-portage/gentoolkit-dev` есть утилита `gensync`, которая поможет поддерживать ваши оверлейные репозитории в актуальном состоянии.

Используя `gensync`, вы можете обновить сразу все репозитории или выбрать для обновления только некоторые из них. В каждом репозитории в каталоге `/etc/gensync/` должен находиться файл `.syncsource`, в котором содержится информация о местоположении репозитория, его имени, идентификаторе и т.д.

Предположим, что у вас есть два дополнительных репозитория с названиями `java` (для сборочных файлов разработок, ведущихся на java) и `entapps` (для внутренних приложений, разработанных на вашем предприятии). Вы можете обновить эти репозитории следующей командой:

Листинг 2.1: Запуск gensync для обновления нескольких репозиториев

```
# gensync java entapps
```

5.c. Программы, поддерживаемые не Portage

Использование Portage с пакетами самостоятельной сборки

Иногда вам может потребоваться сконфигурировать, установить и поддерживать программное обеспечение самостоятельно, без автоматизации со стороны Portage, не смотря на то, что оно поддерживается Portage. Наиболее известные случаи — это исходные коды ядра и драйверы от nVidia. Вы

можете настроить Portage так, чтобы системе стало известно, что определенные пакеты установлены вручную. Этот процесс называется *внедрение*, и поддерживается Portage посредством файла `/etc/portage/profile/package.provided`.

Например, если вы захотите сообщить Portage, что пакет `vanilla-sources-2.6.11.6` установлен вручную, нужно добавить следующую строку в `/etc/portage/profile/package.provided`:

Листинг 3.1: Пример строки из файла package.provided

```
sys-kernel/vanilla-sources-2.6.11.6
```

6. Использование ebuild

6.a. Emerge и Ebuild

Программа `ebuild` — это низкоуровневый интерфейс системы Portage. С ее помощью можно выполнять определенные действия над заданными сборками ebuild. Например, вы можете самостоятельно выполнить отдельные этапы установки.

Программа `ebuild` предназначена в основном для разработчиков, поэтому более подробная информация находится в настольной книге разработчика (англ.). Однако, мы расскажем, какие экземпляры ebuild вызываются системой Portage на разных этапах установки, и как выполнить пост-конфигурационные шаги, которые допускаются некоторыми пакетами.

6.b. Ручная установка программ

Извлечение исходных кодов и проверка контрольных сумм

Каждый раз, когда вы вызываете `ebuild` для какого-то ebuild-файла, проверяется совпадение контрольной суммы всех задействованных файлов с указаной в файлах `Manifest` или `files/digest-<имя>-<версия>`. Проверка выполняется после загрузки исходных кодов.

Чтобы загрузить исходные коды с помощью `ebuild`, запустите:

Листинг 2.1: Загрузка исходных кодов

```
# ebuild путь/к/файлу-ebuild fetch
```

Если контрольная сумма md5 сборочного файла не совпадает с той, что указана в файле `Manifest`, или же один из загруженных файлов не совпадает с описанием в файле `files/digest<пакет>`, вы получите сообщение об ошибке, похожее на такое:

Листинг 2.2: Ошибка контрольной суммы ebuild

```
!!! File is corrupt or incomplete. (Digests do not match)
>>> our recorded digest: db20421ce35e8e54346e3ef19e60e4ee
>>>  your file's digest: f10392b7c0b2bbc463ad09642606a7d6

(!!! Файл поврежден или усечен. (Контрольные суммы не совпадают) )
```

На следующей строке указывается проблемный файл.

Если вы абсолютно уверены, что загруженные исходные коды и сам сборочный файл ebuild именно те, что вам нужны, можете пересоздать файлы `Manifest` и `digest-<пакете>`, используя фукцию digest программы `ebuild`:

Листинг 2.3: Создание новых файлов Manifest и digest

```
# ebuild путь/к/файлу-ebuild digest
```

Распаковка исходных кодов

Чтобы рапаковать исходные коды в `/var/tmp/portage` (или любой другой каталог, указанный в `/etc/make.conf`), запустите функцию unpack программы `ebuild`:

Листинг 2.4: Распаковка исходных кодов

```
# ebuild путь/к/файлу-ebuild unpack
```

Эта команда выполнит функцию src_unpack() программы ebuild (которая по умолчанию просто выполняет распаковку, если функция src_unpack() не определена). Все необходимые заплатки накладываются также на этом этапе.

Компиляция исходных кодов

Следующий шаг в процессе установки — компиляция исходных кодов. Для этого выполняется функция src_compile() вашего сборочного файла. Если нужно, заодно выполняется конфигурация.

Листинг 2.5: Компиляция исходных кодов

```
# ebuild путь/к/файлу-ebuild compile
```

Если вы хотите изменить инструкции компиляции, советуем отредактировать функцию src_compile(). Однако, вы можете также обмануть Portage, заставив ее поверить, что программа `ebuild` уже завершила компиляцию. Запустите нужные команды самостоятельно и создайте пустой файл `.compile` в рабочем каталоге.

Листинг 2.6: Сообщение Portage о завершении задания компиляции

```
# touch .compiled
```

Установка файлов во временное место

Следующий шаг — установка всех необходимых файлов во временный каталог. В него помещаются все файлы, подлежащие включению в рабочую файловую систему. Вы можете выполнить этот этап, запустив функцию установки программы `ebuild`, которая исполняет функцию src_install() сборочного файла.

Листинг 2.7: Установка файлов

```
# ebuild путь/к/файлу-ebuild install
```

Помещение файлов в рабочую файловую систему

Последний этап — перенос всех файлов в рабочую файловую систему и их регистрация в системе Portage. В `ebuild` этот этап называется «qmerge», и включает следующие действия:

- выполняется функция pkg_preinst(), если она определена
- все файлы копируются в рабочую файловую систему
- файлы регистрируются в системе Portage
- выполняется функция pkg_postinst(), если она определена

Запустите функцию qmerge программы `ebuild`, чтобы выполнить этот этап:

Листинг 2.8: Помещение файлов в рабочую файловую систему

```
# ebuild путь/к/файлу-ebuild qmerge
```

Очистка временного каталога

Наконец, можно очистить временный каталог, используя команду clean программы `ebuild`:

Листинг 2.9: Очистка временного каталога

```
# ebuild путь/к/файлу-ebuild clean
```

6.c. Дополнительные возможности Ebuild

Запуск всех команд установки

С помощью функции merge программы `ebuild`, можно запустить команды извлечения, распаковки, компиляции, установки и помещения за один раз:

Листинг 3.1: Установка программы

```
# ebuild путь/к/файлу-ebuild merge
```

Выполнение действий по настройке

В некоторых приложениях содержатся инструкции по дальнейшей настройке установленного пакета. Эти инструкции могут потребовать участия пользователя, и, следовательно, не выполняться автоматически. Для запуска шагов настройки, указанных в необязательной функции config() сборочного файла, используйте команду config программы `ebuild`:

Листинг 3.2: Настройка пакета

```
# ebuild путь/к/файлу-ebuild config
```

Сборка пакета (RPM)

Вы можете попросить Portage создать двоичный пакет или даже RPM из вашего сборочного файла, воспользовавшись командами package и rpm, соответственно. Эти команды несколько различаются:

- команда package во многом похожа на merge, выполняя все необходимые шаги (извлечение, распаковку, компиляцию, установку) перед созданием пакета
- команда rpm собирает пакет RPM из файлов созданных *после* запуска окончания функции install программы ebuild

Листинг 3.3: Создание пакетов

```
(создание двоичного пакета, совместимого с Portage)
# ebuild путь/к/файлу-ebuild package

(создание пакета RPM)
# ebuild путь/к/файлу-ebuild rpm
```

Созданный RPM, однако, не будет содержать информацию о зависимостях из сборочного файла ebuild.

6.d. Дополнительная информация

За дополнительными сведениями о системе Portage, программе ebuild и сценариях ebuild обращайтесь к следующим страницам справки man:

Листинг 4.1: Страницы справки

```
$ man portage     (сама система Portage)
$ man emerge      (команда emerge)
$ man ebuild      (команда ebuild)
$ man 5 ebuild    (синтаксис файлов ebuild)
```

Кроме того, дополнительные сведения, относящиеся к разработке, находятся в [настольной книге]

разработчика (англ.).

D. Настройка сети в Gentoo

1. Начальная настройка

1.a. Приступаем к настройке

Примечание: В документе предполагается, что вы правильно сконфигурировали свое ядро и модули для оборудования, и вам известно интерфейсное имя устройств. Мы также предполагаем, что вы настраиваете `eth0`, хотя на самом деле это может оказаться `eth1`, `wlan0` и т.д.

Примечание: Требуется, чтобы у вас использовался `baselayout-1.11.11` или более свежий.

Для начала настройки своей сетевой платы, нужно рассказать о ней системе Gentoo RC. Это делается созданием символической ссылки с `net.lo` на `net.eth0` в `/etc/init.d`.

Листинг 1.1: Создание символической ссылки с net.lo на net.eth0

```
# cd /etc/init.d
# ln -s net.lo net.eth0
```

Теперь система Gentoo RC знает об этом интерфейсе. Ей также нужно знать, как настраивать новый интерфейс. Конфигурация всех сетевых интерфейсов находится в `/etc/conf.d/net`. Вот простая настройка для использования DHCP или статического адреса.

Листинг 1.2: Примеры для /etc/conf.d/net

```
# использование DHCP
config_eth0=( "dhcp" )

# статический IP-адрес, используется запись CIDR
config_eth0=( "192.168.0.7/24" )
routes_eth0=( "default via 192.168.0.1" )

# статический IP-адрес, запись с маской подсети
config_eth0=( "192.168.0.7 netmask 255.255.255.0" )
routes_eth0=( "default gw 192.168.0.1" )
```

Примечание: Если конфигурация для интерфейса не указывается, предполагается использование DHCP.

Примечание: CIDR расшифровывается как Classless InterDomain Routing (бесклассовая междоменная маршрутизация). Первоначально, адреса IPv4 были разделены на классы A, B и C. Ранняя система классификации не была рассчитана на массовую популярность интернета, и попала под угрозу исчерпания новых уникальных адресов. CIDR — это схема адресации, позволяющая одному IP-адресу обозначать множество IP-адресов. IP-адрес CIDR выглядит как обычный IP-адрес с добавлением косой черты и числа; например, 192.168.0.0/16. CIDR описывается в RFC 1519.

Теперь, настроив интерфейс, мы можем запускать и останавливать его следующими командами:

Листинг 1.3: Сценарии запуска и остановки сети

```
# /etc/init.d/net.eth0 start
# /etc/init.d/net.eth0 stop
```

Важно: При поиске неисправностей сети рекомендуется установить `RC_VERBOSE="yes"` в `/etc/conf.d/rc` для получения более подробной информации о происходящем.

Теперь, успешно запустив и остановив сетевой интерфейс, вы можете захотеть, чтобы он запускался при каждой загрузке Gentoo. Вот как это сделать. Последняя команда «rc» указывает Gentoo, что нужно запускать в текущем уровне запуска любые еще не запущенные сценарии.

Листинг 1.4: Настройка запуска сетевого интерфейса при загрузке

```
# rc-update add net.eth0 default
# rc
```

2. Расширенная настройка

2.a. Расширенная настройка

Переменная `config_eth0` служит основой конфигурации интерфейса. Она содержит список высокоуровневых инструкций по настройке интерфейса (в данном случае, `eth0`). Все команды списка выполняются последовательно. Интерфейс считается работоспособным, если хотя бы одна команда выполнена успешно.

Вот список встроенных инструкций:

Команда	Описание
`null`	**Не выполнять никаких действий**
`noop`	**Если интерфейс включен и существует адрес, успешно завершить настройку.**
an IPv4 or IPv6 address	**Добавить адрес к интерфейсу**
`dhcp`, `adsl` or `apipa` (или команда запуска модуля стороннего изготовителя)	**Запустить модуль, реализующий команду. Например, `dhcp` запускает модуль, реализующий DHCP, которым может быть `dhcpcd`, `udhcpc`, `dhclient` или `pump`.**

На случай неудачного выполнения команды можно указать запасную команду. Запасной вариант должен строго соответствовать структуре конфигурации.

Команды можно сцеплять. Вот несколько практических примеров.

Листинг 1.1: Примеры настройки

```
# Задание трех адресов IPv4
config_eth0=(
  "192.168.0.2/24"
  "192.168.0.3/24"
  "192.168.0.4/24"
)

# Задание одного адреса IPv4 и двух адресов IPv6
config_eth0=(
  "192.168.0.2/24"
  "4321:0:1:2:3:4:567:89ab"
  "4321:0:1:2:3:4:567:89ac"
)

# Сохранять адрес, присвоенный ядром, до отключения интерфейса.
# При этом назначить другой через DHCP. Если DHCP не работает,
# задать статический адрес, определяемый APIPA
config_eth0=(
  "noop"
  "dhcp"
)
fallback_eth0=(
  "null"
  "apipa"
)
```

Примечание: При использовании модуля `ifconfig` для назначения нескольких адресов, для каждого дополнительного адреса создаются псевдонимы интерфейса. Так, в двух примерах, приведенных выше, создаются интерфейсы `eth0`, `eth0:1` и `eth0:2`. С этими интерфейсами нельзя сделать ничего особенного, так как и ядро, и другие программы обрабатывают `eth0:1` и `eth0:2` просто как `eth0`.

Важно: Порядок настройки запасного режима имеет значение! Если бы мы не указали инструкцию `null`, то команда `apipa` запускалась бы только при неудачном выполнении команды `noop`.

Примечание: APIPA и DHCP обсуждаются позже.

2.b. Сетевые зависимости

Сценарии инициализации в `/etc/init.d` могут находиться в зависимости от определенного сетевого интерфейса или просто от службы сети (net). Определив переменную `RC_NET_STRICT_CHECKING` в `/etc/conf.d/rc`, службе `net` можно придать различный смысл.

Значение	Описание
`none`	**Служба `net` считается всегда работающей.**
`no`	**В основном это означает, что по крайней мере одна служба `net.*`, кроме `net.lo`, должна работать. Это может пригодиться пользователям ноутбуков, у которых есть WIFI и статическое проводное подключение, когда нужно, чтобы при включении хотя бы одного интерфейса служба сети выглядела включенной.**
`lo`	**То же, что и `no`, но с учетом `net.lo`. Может быть полезно для тех, кого не волнует, чтобы определенный интерфейс включался при загрузке.**
`yes`	**В этом случае ВСЕ сетевые интерфейсы ДОЛЖНЫ работать, чтобы служба `net` считалась работающей.**

Но как насчет `net.br0`, зависимого от `net.eth0` и `net.eth1`? `net.eth1` может быть беспроводным или PPP-устройством, требующим предварительной настройки для возможности включения в мост. Это невозможно сделать в `/etc/init.d/net.br0`, так как он является символьной ссылкой на `net.lo`.

Ответом является создание своей собственной функции `depend()` в `/etc/conf.d/net`.

Листинг 2.1: Зависимость net.br0 в /etc/conf.d/net

```
# Можно использовать любую зависимость (use, after, before),
# как видно в текущих сценариях
depend_br0() {
  need net.eth0 net.eth1
}
```

Более подробно зависимости обсуждаются в разделе Написание сценариев инициализации Настольной книги Gentoo.

2.c. Имена и значения переменных

Имена переменных являются динамическими. Обычно они следуют структуре `variable_${interface|mac|essid|apmac}`. Например, значение переменной `dhcpcd_eth0` хранит параметры dhcpcd для eth0, а переменной `dhcpcd_essid` — параметры dhcpcd, используемые при подключении любого интерфейса к ESSID «essid».

Однако, не существует твердого простого правила, устанавливающего, что интерфейсы должны называться ethx. На деле, имена многих беспроводных выглядят как wlanx, rax и ethx. Кроме того, некоторые пользовательские интерфейсы, например, мосты, можно называть как угодно, например, foo. Для пущего разнообразия, в именах беспроводных точек доступа также допускаются знаки, не входящие в алфавитно-цифровые; это имеет значение, потому что есть возможность настройки сетевых параметров для отдельных ESSID.

Оборотная сторона всего этого в том, что для настройки сети в Gentoo используются переменные bash, а bash не в состоянии использовать что-либо кроме знаков английского алфавита и цифр. Чтобы обойти такое ограничение, мы заменяем каждый символ, не являющийся английским буквенно-цифровым, на знак подчеркивания: `_`.

Другая особенность bash — это значения переменных: некоторые символы требуют специальной записи, перед ними помещается знак `\`. Им необходимо предварять следующие символы: `"`, `'` и `\`.

В следующем примере мы используем беспроводные ESSID, так как в них может содержаться самое широкое множество символов. Мы воспользуемся ESSID `My "\ NET`:

Листинг 3.1: Пример имени переменной

```
# Этот пример работает, но домен не существует
dns_domain_My____NET="My \"\\ NET"
```

```
# Предыдущая строка устанавливает домен dns в My "\ NET при
# подключении беспроводной платы к точке доступа с ESSID My "\ NET.
```

3. Модульное построение сети

3.a. Сетевые модули

Сейчас мы поддерживаем модульные сетевые сценарии. Это значит, что мы можем легко добавлять поддержку для новых типов интерфейсов и конфигурационных модулей, сохраняя совместимость с существующими.

По умолчанию, модули загружаются только если пакет, нужный им, установлен. Если отметить модуль, для которого не установлен пакет, будет выдана ошибка с указанием, какой пакет нужно установить. В идеале, настройка модулей требуется только тогда, когда установленно несколько пакетов, представляющих одну и ту же службу, а вам установить приоритет одного из них.

Примечание: Все обсуждаемые значения хранятся в `/etc/conf.d/net`, если явно не указано иное.

Листинг 1.1: Предпочтение модуля

```
# выбор iproute2, а не ifconfig
modules=( "iproute2" )

# можно также указать другие модули для отдельного интерфейса
# здесь мы выбираем udhcpc, а не dhcpcd
modules_eth0=( "udhcpc" )

# также можно указать, какие модули не надо использовать: например,
# возможно, вы используете supplicant или linux-wlan-ng для управления
# параметрами беспроводной сети, но при этом желаете настраивать сетевые
# параметры раздельно для каждого связанного ESSID
modules=( "!iwconfig" )
```

3.b. Обработчики интерфейса

Сейчас мы предоставляем два обработчика интерфейса: `ifconfig` и `iproute2`. Для настройки сети вам нужен только один из них.

`ifconfig` в текущем Gentoo используется по умолчанию, и включен в системный профиль. `iproute2` — более мощный и гибкий пакет, который не включен в системный профиль по умолчанию.

Листинг 2.1: Установка iproute2

```
# emerge sys-apps/iproute2

# выбор iproute2, а не ifconfig, когда установлены оба
modules=( "iproute2" )
```

Так как и `ifconfig` и `iproute2` делают очень сходные вещи, то мы сделали их базовую настройку взаимозаменяемой. Например, оба приведенных ниже примера работают не зависимо от того, какой модуль используется.

Листинг 2.2: Примеры ifconfig и iproute2

```
config_eth0=( "192.168.0.2/24" )
config_eth0=( "192.168.0.2 netmask 255.255.255.0" )

# также можно указать широковещательный адрес
config_eth0=( "192.168.0.2/24 brd 192.168.0.255" )
config_eth0=( "192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255" )
```

3.c. DHCP

DHCP — это способ получения сетевой информации (адреса IP, сервера DNS, шлюза и т.д.) с сервера. Это значит, что если в сети запущен сервер DHCP, вам остается только сказать каждому клиенту, чтобы он использовал DHCP, и сеть настроится сама собой. Конечно, вам придется настраивать все остальное (бесроводную сеть, подключение точка-точка и т.д.), если они должны работать до использования DHCP.

Поддержка DHCP обеспечивается `dhclient`, `dhcpcd`, `pump` или `udhcpc`. У каждого модуля DHCP есть свои плюсы и минусы: здесь мы быстренько рассмотрим их.

Модуль DHCP	Пакет	Плюсы	Минусы
`dhclient`	`net-misc/dhcp`	**Сделан ISC, теми же людьми, кто делает BIND DNS. Гибок в настройке.**	**Настройка чрезмерно сложна, программа довольно «распухшая», не может получать данные о серверах NTP с DHCP, по умолчанию не отправляет имя узла.**
`dhcpcd`	`net-misc/dhcpcd`	**Давно в Gentoo по умолчанию, не зависит от внешних утилит.**	**Более не поддерживается разработчиком, может быть временами медленным, не становится демоном при неограниченном сроке аренды адреса.**
`pump`	`net-misc/pump`	**Компактный, не зависит от внешних утилит.**	**Более не поддерживается разработчиком, ненадежен, особенно по модему, не может получать данные о серверых NIS по DHCP.**
`udhcpc`	`net-misc/udhcp`	**Компактный; наименьший существующий клиент DHCP, сделан для встроенных систем.**	**Не зарекомендовал себя — ни в одном дистрибутиве не используется по умолчанию; не поддерживает длительность таймаута более 3 секунд.**

Если у вас установлено больше одного DHCP клиента, вам нужно указать, какой использовать; иначе по умолчанию используется `dhcpcd`, если есть.

Чтобы передать определенные параметры модулю DHCP, используйте `модуль_eth0="..."` *(замените модуль на имя используемого модуля DHCP, например, `dhcpcd_eth0`)*.

Мы попытались сделать DHCP относительным агностиком: по существу, мы поддерживаем следующие команды, с использованием переменной `dhcp_eth0`. По умолчанию не включена ни одна из них.

- `release` — освобождать IP-адрес для повторного использования
- `nodns` — не замещать `/etc/resolv.conf`
- `nontp` — не замещать `/etc/ntp.conf`
- `nonis` — не замещать `/etc/yp.conf`

Листинг 3.1: Простая настройка DHCP в /etc/conf.d/net

```
# требуется только если у вас несколько модулей DHCP
modules=( "dhcpcd" )

config_eth0=( "dhcp" )
dhcpcd_eth0="-t 10" # прекращение после 10 секунд
dhcp_eth0="release nodns nontp nonis" # только получать адрес
```

Примечание: По умолчанию, `dhcpcd`, `udhcpc` и `pump` передают текущее узла на сервер DHCP, поэтому его больше не требуется указывать.

3.d. Модем ADSL

Сначала нужно установить программное обеспечение для ADSL.

Листинг 4.1: Установка пакета rp-pppoe

```
# emerge net-dialup/rp-pppoe
```

Предупреждение: В `baselayout-1.11.x` поддерживается только PPPoE. Надеемся, что в будущих версиях появится поддержка PPPoA.

Сейчас нам нужно указать, что на `eth0` будет ADSL-интерфейс, и ввести наше имя пользователя, обновив `/etc/conf.d/net`.

Листинг 4.2: Настройка eth0 для ADSL в /etc/conf.d/net

```
config_eth0=( "adsl" )
adsl_user_eth0="имя-пользователя"
```

Наконец, нужно указать ваше имя и пароль в `/etc/ppp/pap-secrets`.

Листинг 4.3: Пример /etc/ppp/pap-secrets

```
# * обязательна
"пользователь"  *  "пароль"
```

3.e. APIPA (автоматическая частная IP-адресация)

APIPA пытается найти свободный адрес в диапазоне 169.254.0.0-169.254.255.255, проверяя отклик на интерфейсе произвольного адреса из этого диапазона по протоколу arp. Если отклика нет, адрес назначается интерфейсу.

Это полезно только в локальных сетях, где нет сервера DHCP, нет прямого подключения к интернету, и все другие компьютеры используют APIPA.

Для поддержки APIPA установите `net-misc/iputils` или `net-analyzer/arping`.

Листинг 5.1: Настройка APIPA в /etc/conf.d/net

```
# сначала пробуем DHCP, при неудаче переходим на APIPA
config_eth0=( "dhcp" )
fallback_eth0=( "apipa" )

# использование только APIPA
config_eth0=( "apipa" )
```

3.f. Объединение интерфейсов

Для объединения каналов в ствол (bonding) установите `net-misc/ifenslave`.

Объединение используется для повышения пропускной способности сети. Если у вас есть две сетевых карты, выходящих в одну и ту же сеть, можно объединить их, так что ваши приложения увидят только один интерфейс, но реально будут пользоваться двумя сетевыми платами.

Листинг 6.1: Настройка объединения в /etc/conf.d/net

```
# объединение интерфейсов
slaves_bond0="eth0 eth1 eth2"

# вы можете не захотеть назначать адрес IP объединенному интерфейсу
config_bond0=( "null" )

# указание зависимости от eth0, eth1 и eth2, так как им может требоваться
# дополнительная настройка
depend_bond0() {
need net.eth0 net.eth1 net.eth2
}
```

3.g. Образование моста (поддержка 802.1d)

Для поддержки мостов установите `net-misc/bridge-utils`.

Мосты используются для объединения сетей. Например, у вас может быть сервер, подсоединенный к интернету через ADSL-модем, и плата беспроводного доступа для предоставления доступа в интернет через ADSL модем другим компьютерам. Чтобы соединить оба интерфейса, можно создать «мост».

Листинг 7.1: Настройка моста в /etc/conf.d/net

```
# настройка моста: подробности в "man btctl"
brctl_br0=( "setfd 0" "sethello 0" "stp off" )

# включаем порты в мост br0
bridge_br0="eth0 eth1"

# устанавливаем порты в "null", чтобы не запускался dhcp
config_eth0=( "null" )
config_eth1=( "null" )

# наконец, даем мосту адрес; можно использовать и DHCP
config_br0=( "192.168.0.1/24" )

# указываем зависимость от eth0 и eth1, так как им может требоваться
# дополнительная настройка
depend_br0() {
need net.eth0 net.eth1
}
```

Важно: Для использования некоторых вариантов моста вам может потребоваться обращение к документации по именам переменных.

3.h. MAC-адрес

Для изменения MAC-адреса своего интерфейса вам не потребуется ничего устанавливать, если у вас `sys-apps/baselayout-1.11.14` или новее, и вы собираетесь сменить MAC-адрес на какой-то определенный. Однако, если вам нужно сменить MAC-адрес на случайный, или ваш baselayout старше указанной версии, для смены адреса потребуется установить пакет командой emerge `net-analyzer/macchanger`.

Листинг 8.1: Пример изменения MAC-адреса

```
# установка MAC-адреса интерфейса
mac_eth0="00:11:22:33:44:55"

# случайная установка последних 3 байт адреса
mac_eth0="random-ending"

# установка случайного адреса из диапазона для физического соединения
# того же типа (оптического, медного, беспроводного) любого изготовителя
mac_eth0="random-samekind"

# установка случайного адреса из диапазона для любого типа физического
# соединения (оптического, медного, беспроводного) любого изготовителя
mac_eth0="random-anykind"

# полностью случайный; ВНИМАНИЕ, некоторые MAC-адреса, сгенерированные
# таким образом, могут вести себя НЕ ТАК, как предполагается
mac_eth0="random-full"
```

3.i. Образование тоннеля

Для образования тоннеля вам не нужно ничего устанавливать, поскольку на это способен сам обработчик интерфейса.

Листинг 9.1: Настройка тоннеля в /etc/conf.d/net

```
# для тоннелей GRE
iptunnel_vpn0="mode gre remote 207.170.82.1 key 0xffffffff ttl 255"

# для тоннелей IPIP
iptunnel_vpn0="mode ipip remote 207.170.82.2 ttl 255"

# для настройки интерфейса
config_vpn0=( "192.168.0.2 peer 192.168.1.1" )
```

3.j. Виртуальные сети (поддержка 802.1q)

Для поддержки VLAN, установите `net-misc/vconfig`.

Виртуальная локальная сеть (VLAN) — это группа сетевых устройств, которые ведут себя, как будто подключены к одному сегменту сети, даже когда это не так. Членам VLAN видны только члены той же VLAN даже если в той же физической сети присутствуют другие.

Листинг 10.1: VLAN configuration in /etc/conf.d/net

```
# указание номеров VLAN для интерфейса
# пожалуйста, убедитесь, что ваши номера VLAN НЕ дополнены нулем
vlans_eth0="1 2"

# можно также настроить VLAN
# за подробностями обращайтесь к man vconfig
vconfig_eth0=( "set_name_type VLAN_PLUS_VID_NO_PAD" )
vconfig_vlan1=( "set_flag 1" "set_egress_map 2 6" )

# настройка интерфейса как обычно
config_vlan1=( "172.16.3.1 netmask 255.255.254.0" )
config_vlan2=( "172.16.2.1 netmask 255.255.254.0" )
```

Важно: Для использования некоторых вариантов VLAN вам может потребоваться обращение к документации по именам переменных.

4. Беспроводная сеть

4.a. Введение

В настоящее время поддерживается подключение к беспроводной сети с помощью `wireless-tools` или `wpa_supplicant`. Важно помнить, что подключение к беспроводным сетям настраивается глобально, а не для определённого интерфейса.

`wpa_supplicant` — лучший выбор, но он поддерживает не все драйверы. Список поддерживаемых драйверов находится на сайте wpa_suppliant. Кроме того, сейчас `wpa_supplicant` может подключаться только к тем сетям, на SSID которых настроен.

`wireless-tools` поддерживает практически все платы и драйверы, но не способен подключаться к точкам доступа, работающим исключительно с WPA.

Предупреждение: Драйвер `linux-wlan-ng` в данный момент не поддерживается в baselayout. Это из-за того, что в `linux-wlan-ng` своя собственная программа установки и настройки, которая ни на что не похожа. Разработчики `linux-wlan-ng`, по слухам, собираются перейти на установку как в `wireless-tools`; когда это произойдет, вы сможете использовать `linux-wlan-ng` с baselayout.

4.b. Запросчик WPA

Запросчик WPA (WPA Supplicant) — пакет, позволяющий подсоединяться к точкам доступа с протоколом WPA. Его настройка проходит достаточно гладко, и пакет работает достаточно стабильно, хотя находится на стадии бета-тестирования.

Листинг 2.1: Установка wpa_supplicant

```
# emerge net-wireless/wpa_supplicant
```

Важно: Для работы `wpa_supplicant` в ядре должен быть включен параметр `CONFIG_PACKET`.

Теперь нам необходимо настроить `/etc/conf.d/net` для предпочтения `wpa_supplicant` по отношению к `wireless-tools` (по умолчанию, если обе программы установлены, работает `wireless-tools`).

Листинг 2.2: Настройка /etc/conf.d/net для wpa_supplicant

```
# выбор wpa_supplicant
modules=( "wpa_supplicant" )

# важно указать wpa_supplicant, какой драйвер нужно использовать,
# так как программа пока не слишком хорошо угадывает сама

wpa_supplicant_eth0="-Dбезумный-wifi"
```

Примечание: Если вы используете драйвер host-ap, то вам потребуется перевести плату в *ведомый режим (managed mode)*, прежде чем она сможет правильно работать с `wpa_supplicant`. Для этого можно указать `iwconfig_eth0="mode managed"` в `/etc/conf.d/net`.

Это довольно просто, не так ли? Однако, нужно настроить саму программу `wpa_supplicant`, что значительно сложнее. Сложность зависит от степени защиты точек доступа, к которым вы собираетесь подключаться. Следующий упрощенный пример взят из `/etc/wpa_supplicant.conf.example`, поставляемого в составе `wpa_supplicant`.

Листинг 2.3: Пример /etc/wpa_supplicant.conf

```
# следующую строку нельзя изменять, иначе программа не сможет работать
ctrl_interface=/var/run/wpa_supplicant

# ограничим доступ к настройкам WPA только для root
ctrl_interface_group=0

# пусть wpa_supplicant заботится о сканировании и выборе точки доступа
ap_scan=1

# простой случай: WPA-PSK, согласованный ключ - текстовая строка,
# принимать любой допустимый шифр
network={
  ssid="просто"
  psk="очень тайный пароль"
  # чем выше приоритет, тем скорее выбор
  priority=5
}

# как в предыдущем, но с запросом сканирования по определенному SSID
# (для точек доступа, отклоняющих широковещательный SSID)
network={
  ssid="второй ssid"
  scan_ssid=1
  psk="очень тайный пароль"
  priority=2
}

# использовать только WPA-PSK; принимать любое допустимое сочетание шифров

network={
  ssid="пример"
  proto=WPA
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP WEP104 WEP40
  psk=06b4be19da289f475aa46a33cb793029d4ab3db7a23ee92382eb0106c72ac7bb
  priority=2
}

# открытое подключение без шифрования (не WPA, не IEEE #802.1X)

network={
  ssid="тест-открытого-текста"
  key_mgmt=NONE
}

# подключение с общим ключом WEP (не WPA, не IEEE #802.1X)
network={
  ssid="тест-статического-wep"
  key_mgmt=NONE
  wep_key0="abcde"
```

```
  wep_key1=0102030405
  wep_key2="1234567890123"
  wep_tx_keyidx=0
  priority=5
}

# подключение с общим ключом WEP (не WPA, не IEEE #802.1X),
# допуск с использованим общего ключа IEEE 802.11
network={
  ssid="тест2-статического-wep"
  key_mgmt=NONE
  wep_key0="abcde"
  wep_key1=0102030405
  wep_key2="1234567890123"
  wep_tx_keyidx=0
  priority=5
  auth_alg=SHARED
}

# сеть IBSS/ad-hoc с WPA-None/TKIP
network={
  ssid="тест adhoc"
  mode=1
  proto=WPA
  key_mgmt=WPA-NONE
  pairwise=NONE
  group=TKIP
  psk="тайный пароль"
}
```

4.c. Утилиты Wireless tools

Начальная установка и режим ведомого

Wireless Tools обеспечивают общий способ настройки базовых беспроводных интерфейсов, вплоть до защиты WEP. Хотя WEP является слабым методом защиты, он наиболее распространен.

Для настройки Wireless Tools служат несколько основных переменных. В примере файла настроек, приведенном ниже, описано все, что вам потребуется. Нужно помнить, что отсутствие настройки означает «подключаться к нешифрующей точке доступа с самым сильным сигналом». Программа будет всегда пытаться подключить вас к чему-нибудь.

Листинг 3.1: Установка wireless-tools

```
# emerge net-wireless/wireless-tools
```

Примечание: Хотя вы можете хранить свои параметры настройки беспроводной сети в `/etc/conf.d/wireless`, это руководство рекомендует хранить их в `/etc/conf.d/net`.

Важно: Вам *понадобится* обратиться к документации по именам переменных.

Листинг 3.2: Пример настройки iwconfig /etc/conf.d/net

```
# приоритет использования iwconfig над wpa_supplicant
modules=( "iwconfig" )

# Настройка ключей WEP для точек доступа ESSID1 и ESSID2
# Можно указывать до 4 ключей WEP, но только 1 может работать в каждый
# момент, поэтому мы указываем индекс по умолчанию [1], чтобы сделать ключ [1],
# а впоследствии снова, чтобы изменить активный ключ на [1].
# Это нужно, если вы настраиваете другие ESSID на использование WEP-ключей,
# отличающихся от [1].
#
# Приставка s: перед ключом означает, что ключ текстовый, иначе -
# шестнадцатиричный
#
# enc open указывает открытую защиту (более безопасно)
# enc restricted указывает ограниченную защиту (менее безопасно)
key_ESSID1="[1] s:ваш-ключ-здесь key [1] enc open"
```

```
key_ESSID2="[1] aaaa-bbbb-cccc-dd key [1] enc restricted"
#
# Нижеследующее работает только при поиске доступных точек доступа.

# Иногда видны несколько точек доступа, и требуется задать
# предпочтительный порядок подключения
preferred_aps=( "ESSID1" "ESSID2" )
```

Настройка порядка выбора точки доступа

Можно указать несколько дополнительных параметров для уточнения порядка выбора точки доступа, но обычно этого не требуется.

Вам решать, подключаться ли только к указанным точкам доступа, или нет. По умолчанию, если подключение ко всем настроенным точкам доступа не удалось, и есть возможность подключиться к нешифрующей точке доступа, такое соединение произойдет. Этот порядок зависит от переменной `associate_order`. Ниже приводится таблица значений и с описанием их действия.

Значение	Описание
`any`	**поведение по умолчанию**
`preferredonly`	**соединяться только с видимыми точками доступа из списка**
`forcepreferred`	**принудительно подключаться к точкам доступа в заданной последовательности, если они не обнаружены при сканировании**
`forcepreferredonly`	**не сканировать точки доступа, просто пытаться подключиться к каждой по списку**
`forceany`	**так же, как в `forcepreferred` + подключаться к любой доступной точке доступа**

Наконец, мы можем указать `blacklist_aps` и `unique_ap`. `blacklist_aps` работает подобно `preferred_aps`. `unique_ap` устанавливается в `yes` или `no`, указывая, можно ли подключать второй беспроводной интерфейс к той же точке доступа, что и первый.

Листинг 3.3: Пример blacklist_aps и unique_ap

```
# иногда требуется полностью исключить возможность подключения
# к определенным точкам доступа
blacklist_aps=( "ESSID3" "ESSID4" )

# если у вас несколько беспроводных плат, можно указать, можно ли им
# подключаться к одной и той же точке доступа
# значение - "yes" или "no"
# по умолчанию - "yes"
unique_ap="yes"
```

Режим отдельного и ведущего узла

Если вы хотите становиться отдельным узлом (ad hoc), когда не удается подключиться ни к какой точке доступа в ведомом режиме, это тоже возможно.

Листинг 3.4: Откат к режиму отдельного узла

```
adhoc_essid_eth0="Этот отдельный узел"
```

Как насчет подключения к сетям Ad-Hoc или запуска в режиме ведущего (master), чтобы стать точкой доступа? Есть конфигурация и для такой работы! Вам может потребоваться определить WEP-ключи, как показано выше.

Листинг 3.5: Пример настройки ad-hoc/master

```
# установка режима: допускается managed (ведомый, по умолчанию),
# ad-hoc (отдельный) или или master (ведущий). Не все драйверы поддерживают
# каждый режим
mode_eth0="ad-hoc"

# установка ESSID интерфейса
# в ведомом режиме заставляет интерфейс пытаться подключиться к указанному
```

```
# ESSID, и больше ничего
essid_eth0="Этот отдельный узел"

# если не указан, используется канал 3
channel_eth0="9"
```

Важно: Следующий текст взят дословно из документации BSD wavelan, входящей в документацию NetBSD. «Существуют 14 каналов. Нам сообщили, что использование каналов с 1 по 11 является законным в Северной Америке, каналов с 1 по 13 — в большинстве стран Европы, каналов с 10 по 13 — во Франции, и только канала 14 — в Японии. Если у вас есть сомнения, обратитесь к документации от вашей платы или точки доступа. Убедитесь что выбранный канал совпадает с каналом точки доступа (или другой платы в сети ad-hoc). По умолчанию на платах, продаваемых в Северной Америке и большинстве стран Европы, настроен канал 3; на платах, продаваемых во Франции — канал 11; на платах, продаваемых в Японии — канал 14.»

Устранение неполадок в wireless tools

Существуют дополнительные переменные, которые можно использовать для запуска своего беспроводного оборудования и устранения неполадок, возникших из-за драйвера или проблем с сетевым окружением. Ниже приведена таблица прочих функций, которые можно перепробовать.

Переменная	Значение по умолчанию	Описание
`iwconfig_eth0`		**За подробными сведениями о параметрах `iwconfig` обращайтесь к странице справки iwconfig.**
`iwpriv_eth0`		**За подробными сведениями о параметрах `iwpriv` обращайтесь к странице справки iwpriv**
`sleep_scan_eth0`	`0`	**Время задержки в секундах перед попыткой сканирования. Требуется, когда драйверу или прошивке нужно дополнительное время для перехода в рабочий режим.**
`sleep_associate_eth0`	`5`	**Время ожидания связи интерфейса с точкой доступа (в секундах) перед переходом к опросу следующей.**
`associate_test_eth0`	`MAC`	**Некоторые драйверы не сбрасывают MAC-адрес, связанный с недоступной точкой доступа, при потере или попытки связи. Некоторые драйверы не сбрасывают значение качества сигнала при потере или попытке соединения. Допустимые значения: `MAC`, `quality` и `all`.**
`scan_mode_eth0`		**Некоторым драйверам необходимо сканировать в режиме ad-hoc. Если сканирование не удается, попробуйте указать здесь `ad-hoc`.**
`iwpriv_scan_pre_eth0`		**Посылать интерфейсу некоторые команды `iwpriv` перед сканированием. За дополнительными сведениями обращайтесь к странице справки iwpriv.**
`iwpriv_scan_post_eth0`		**Посылать интерфейсу некоторые команды `iwpriv` после сканирования. За дополнительными сведениями обращайтесь к странице справки iwpriv.**

4.d. Раздельная настройка сети по ESSID

Иногда необходим статический IP при соединении с `ESSID1`, и DHCP при соединении с `ESSID2`. На деле, большинство переменных модуля можно определять раздельно по ESSID. Вот как это сделать:

Примечание: Это работает при использовании WPA Supplicant или Wireless Tools.

Важно: Вам *потребуется* свериться с документацией по именам переменных.

Листинг 4.1: Назначение сетевых настроек для ESSID

```
config_ESSID1=( "192.168.0.3/24 brd 192.168.0.255" )
routes_ESSID1=( "default via 192.168.0.1" )

config_ESSID2=( "dhcp" )
fallback_ESSID2=( "192.168.3.4/24" )
fallback_route_ESSID2=( "default via 192.168.3.1" )

# можно также указать сервера имен и др.
# ПРЕДУПРЕЖДЕНИЕ: DHCP переопределит настройки, если не указано иное
dns_servers_ESSID1=( "192.168.0.1" "192.168.0.2" )
dns_domain_ESSID1="some.domain"
dns_search_domains_ESSID1="search.this.domain search.that.domain"

# перенастройка производится по MAC-адресу точки доступа;
# это полезно, когда в разных местах есть точки доступа с одинаковым ESSID
```

```
config_001122334455=( "dhcp" )
dhcpcd_001122334455="-t 10"
dns_servers_001122334455=( "192.168.0.1" "192.168.0.2" )
```

5. Дополнительные возможности

5.a. Стандартные функции-обработчики

Можно определить четыре функции, которые вызываются при операциях запуска (`start`) и останова (`stop`). При вызове функциям передается название интерфейса, так что одна и та же функция может управлять несколькими адаптерами.

Для указания на то, что запуск или останов интерфейса может продолжаться, возвращаемое значение функций `preup()` и `predown()` должно быть нулевым (успешным). Если `preup()` возвращает ненулевое значение, запуск интерфейса прерывается. Если `predown()` возвращает ненулевое значение, не допускается продолжение останова интерфейса.

Возвращаемое значение функций `postup()` и `postdown()` игнорируется, так как показываемая ими ошибка не обрабатывается.

`${IFACE}` присваивается название запускаемого/останавливаемого интерфейса. `${IFVAR}` — это значение `${IFACE}`, преобразованное в имя переменной, разрешенное в bash.

Листинг 1.1: Примеры функций до/после запуска/останова

```
preup() {
  # Проверка соединения интерфейса перед его запуском. Она
  # работает лишь с некоторыми сетевыми адаптерами и требует наличия
  # установленного пакета mii-diag.
  if mii-tool ${IFACE} 2> /dev/null | grep -q 'no link'; then
    ewarn "Интерфейс ${IFACE} не подключен, прерывание запуска"
    return 1
  fi

  # Проверка соединения интерфейса перед его запуском. Она
  # работает лишь с некоторыми сетевыми адаптерами и требует наличия
  # установленного пакета ethtool.
  if ethtool ${IFACE} | grep -q 'Link detected: no'; then
    ewarn "Интерфейс ${IFACE} не подключен, прерывание запуска"
    return 1
  fi

  # Не забываем вернуть 0 при успехе
  return 0
}

predown() {
  # Назначение этого сценария - проверить наличие корня NFS
  # и в этом случае предотвратить останов интерфейсов. Заметьте, что
  # определяя функцию predown(), вы отменяете существующую логику.
  # Вот она, на случай если все же понадобится...
  if is_net_fs /; then
    eerror "Корневая ФС смонтирована в сети - останов ${IFACE} невозможен"
    return 1
  fi

  # Не забываем вернуть 0 при успехе
  return 0
}

postup() {
  # Эту функцию можно использовать, например, для регистрации в
  # службе динамического DNS. Другой пример - отправка/прием почты после
  # запуска интерфейса.
       return 0
}

postdown() {
```

```
  # Эта функция приводится в основном для полноты... Я не придумал,
  # что бы ценное в нее поместить ;-)
  return 0
}
```

5.b. Функции-обработчики wireless tools

Примечание: Это не работает вместе с WPA Supplicant, но переменные `${ESSID}` и `${ESSIDVAR}` доступны в функции `postup()`.

Можно определить две функции, вызываемые до и после функции подключения (associate). При вызове им сначала передается название интерфейса, так что одна и та же функция может управлять несколькими адаптерами.

Для указания на то, что запуск или останов интерфейса можно продолжать, возвращаемое значение функции `preassociate()` должно быть нулевым (успешным). Если `preassociate()` возвращает ненулевое значение, запуск интерфейса прерывается.

Возвращаемое значение функции `postassociate()` игнорируется, так как показываемая ей ошибка не обрабатывается.

`${ESSID}` присваивается точный ESSID точки доступа, к которой вы подключаетесь. `${ESSIDVAR}` — это `${ESSID}`, преобразованный в имя переменной, разрешенное в bash.

Листинг 2.1: Функции до/после соединения

```
preassociate() {
  # Ниже добавляются две конфигурационных переменных, leap_user_ESSID
  # и leap_pass_ESSID. Когда они обе настроены на подключаемый ESSID,
  # мы запускаем сценарий CISCO LEAP

  local user pass
  eval user=\"\$\{leap_user_${ESSIDVAR}\}\"
  eval pass=\"\$\{leap_pass_${ESSIDVAR}\}\"

  if [[ -n ${user} && -n ${pass} ]]; then
    if [[ ! -x /opt/cisco/bin/leapscript ]]; then
      eend "Для поддержки LEAP, выполните emerge net-misc/cisco-aironet-client-utils"
      return 1
    fi
    einfo "Ожидание допуска LEAP на \"${ESSID//\\\\\//}\""
    if /opt/cisco/bin/leapscript ${user} ${pass} | grep -q 'Login incorrect'; then
      ewarn "Вход пользователя ${user} не удался"
      return 1
    fi
  fi

  return 0
}

postassociate() {
  # Эта функция приводится в основном для полноты... Я не придумал,
  # что бы ценное в нее поместить ;-)

  return 0
}
```

Примечание: `${ESSID}` и `${ESSIDVAR}` недоступны в функциях `predown()` и `postdown()`.

6. Управление сетью

6.a. Управление сетью

Если вы часто берете компьютер в поездки, то у вас не всегда может быть возможность подключения к

сети Ethernet или к беспроводной точке доступа. Но мы можем захотеть, чтобы сеть заработала автоматически, как только к компьтеру подключен кабель Ethernet или найдена беспроводная точка доступа.

Здесь вы найдете некоторые инструменты, которые помогут это организовать.

Примечание: В этом документе рассказывается только о `ifplugd`, но есть и альтернативные решения, например, `quickswitch`.

6.b. ifplugd

ifplugd — это программа, которая запускает и останавливает интерфейс при подключении или отключении кабеля к сети Ethernet. Также она может обработать подключение к беспроводной точке доступа или появление новых точек доступа.

Листинг 2.1: Установка ifplugd

```
# emerge sys-apps/ifplugd
```

Настройка ifplugd — довольно простая задача. Файл конфигурации расположен по адресу: `/etc/conf.d/ifplugd`. Для просмотра подробного описания переменных запустите `man ifplugd`.

Листинг 2.2: Пример конфигурации ifplug

```
# назначение интерфейса для слежения
INTERFACES="eth0"

AUTO="no"
BEEP="yes"
IGNORE_FAIL="yes"
IGNORE_FAIL_POSITIVE="no"
IGNORE_RETVAL="yes"
POLL_TIME="1"
DELAY_UP="0"
DELAY_DOWN="0"
API_MODE="auto"
SHUTDOWN="no"
WAIT_ON_FORK="no"
MONITOR="no"
ARGS=""

# дополнителные параметры ifplugd для указанного интерфейса.
# учтите, установки глобальных переменных игнорируются, если указаны значения
# для конкретного интерфейса
MONITOR_wlan0="yes"
DELAY_UP_wlan0="5"
DELAY_DOWN_wlan0="5"
```