МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Операционные системы»

**Лабораторная работа**

Изучение дистрибутива Linux Gentoo

Выполнили:

студенты группы ИВТАПбд-31

Кондратьев П. С.

Аввакумова И.И.

Проверила:

Валюх В. В.

Ульяновск, 2018

Содержание

Введение…………………………………………………………………………3

1. Техническое задание…………………………………………………………..4

2. Анализ предметной области……………………………………......................6

3. Архитектура ОС………………………………..................................................9

4. Установка дистрибутива Linux Gentoo…………………………………...…14

4.1. Подготовка дисков………………………………………………..….16

4.2. Установка установочных файлов Gentoo…………….......................21

4.3. Установка базовой системы Gentoo……………...............................23

4.4. Настройка ядра Linux……………......................................................28

4.5. Настройка параметров системы…………….....................................30

4.6. Установка системных средств……………........................................31

4.7. Настройка начального загрузчика. . . . . . . . . . . . . . . . . . . . .. .. . .. .33

4.8. Завершение установки……………....................................................34

5.Установска графической оболочки Xfce ………...........................................36

6. Изучение и интеграция модуля в выбранную ОС…………....…….…...….40

Заключение……………………………………………………….…...…....…....50

Используемые ресурсы……………………………...………...……….....……..51

**Введение**

Linux - семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе как правило создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов - в форме, готовой для установки и удобной для сопровождения и обновлений, - и имеющих свой набор системных и прикладных компонентов, как свободных, так возможно и собственнических.

Существует огромное количество диструбутивов Linux, каждая из которых предлагает то или иное преимущество. Различные варианты ОС могут затруднить выбор и забрать немалое количество времени для поиска необходимой именно под ваши задачи. Ubuntu, Mint, Elementary, Fedora, OpenSUSE, Gentoo… список удобных для пользователя дистрибутивов может показаться бесконечной историей.

GNU/Linux сделали по образцу ОС Unix. С самого начала эта операционная система разрабатывалась как многопользовательская и многозадачная. Уже этого достаточно, чтобы выделять её. Но отличий в ней значительно больше. Самое главное – это бесплатность (значительная часть разработок была создана добровольцами на безвозмездной основе) и отсутствие владельца.

Он был создан и продолжает совершенствоваться и развиваться группой добровольцев, первоначально в кругу пользователей сети Internet, которые обменивались кодами, информацией об обнаруженных ошибках, выявлением проблем, возникавших при расширении сферы применения. Все желающие приглашаются подключиться к этой работе. Единственное, что требуется - это интерес к семейству UNIX и желание совершенствовать свои навыки в этой сфере.

**1. Техническое задание**

В рамках лабораторных работ необходимо реализовать большое исследование (исследовательскую лабораторную работу).

В данной лабораторной работе мы исследуем внутреннюю архитектуру операционных систем, погружаемся внутрь различных систем и исследуем отдельные модули.

Исследовательская работа делаются рабочими группами по 2-3 человека (группы формируются из ИВТВМбд и ИВТАПбд самостоятельно студентами, АП и ВМ можно смешивать).

**Задание №1:**

* Необходимо сформировать рабочую группу 2-3 человека, прислать рабочую группу и выбранную ОС(дистрибутив Linux) преподавателю.
* Необходимо выбрать для обзора любой из современных дистрибутивов. При выборе требуется учитывать, что ОС должна быть с открытым исходным кодом.

Выбранная ОС закрепляется за рабочей группой и другим уже нельзя её использовать.

Что входит в обзор:

* разбор архитектуры ОС (прямо схема, рисунок из блоков) с описанием на 2-5 страниц;
* описание отдельных модулей:
* название;
* за что отвечает, функционал;
* структура входных и выходных данных;
* как разрабатывается: с какого года, сколько лет, коллектив разработчиков;
* Финансы: бюджет разработки ОС, кто инвестирует, какова финансовая отдача, на какие средства ведётся разработка.
* список источников.

**Задание №2:**

Выбрать любой модуль, который соответствует выбранному дистрибутиву (согласовать с преподавателем).

Изучить этот модуль, подготовить описание этого модуля.

Что входит в описание модуля:

* схема взаимодействия с другими модулями операционной системы;
* функционал модуля;
* частичный обзор кода.

**Задание №3:**

* Выбирать любой новый модуль, который можно установить в выбранный дистрибутив (согласовать с преподавателем).
* Установить его в ОС.

Возможно потребуется перекомпиляция.

**Задание №4:**

Добавить в код любого модуля новую функциональность (сохранить старую и добавить новую).

Потребуется перекомпиляция.

Сдача лабораторной работы:

* Презентация.
* Отчёт на 20-45 страниц.
* Защита с демонстрацией операционной системы.
* Обязательное присутствие всей рабочей группы.

**2. Анализ предметной области**

Предметной областью является изучение основных особенностей дистрибутива Gentoo, его особенностей модулей и рабочего функционала.

Gentoo Linux ведёт своё начало с разработки Дэниелом Роббинсом дистрибутива Enoch Linux в 1999 году. Благодаря системе Portage, позволяющей сделать из Gentoo практически всё, что угодно (от сервера до рабочей станции), этот дистрибутив можно отнести к так называемым мета-дистрибутивам.

Философия Gentoo проистекает от дней основания дистрибутива и остаётся в неизменном виде и по сей день. Вот как вкратце излагает концепцию проекта его основатель Дэниел Роббинс:

Каждому пользователю приходится выполнять определённую работу. Цель Gentoo — разработка инструментов и систем, позволяющих пользователю заниматься своим делом как можно эффективнее и в своё удовольствие, так, как он сочтёт нужным. Наши инструменты должны приносить радость и помогать пользователю оценить по достоинству всё богатство Linux и сообщества свободного программного обеспечения, а также гибкость свободных программ. Такое возможно только тогда, когда инструменты создаются, чтобы отражать и проводить волю пользователя, оставляя для него открытыми все возможности вплоть до финальной стадии (исходного кода). Когда инструмент заставляет пользователя действовать определённым образом, инструмент работает против него, а не на него. Все мы сталкивались с ситуациями, когда инструменты стремились навязать нам свою собственную волю. Такое положение дел — шаг назад, несовместимый с философией Gentoo.

Иными словами, философия Gentoo — создание лучших инструментов. Когда инструмент в совершенстве выполняет свою задачу, вы можете даже не замечать его присутствия, потому что он не перечит вам, не проявляет себя и не заставляет вас возиться с ним, когда вам совершенно не до этого. Инструмент служит пользователю, а не пользователь — инструменту.

Будущая задача Gentoo — продолжать борьбу за создание инструментов, близких к идеалу. Инструментов, удовлетворяющих нужды множества различных пользователей (каждого — со своими разнообразными целями) с простотой, идущей рука об руку с непревзойдённой мощью.

Таким образом к основным особенностям genntoo можно отнести:

* Мощная и гибкая технология [Portage](https://ru.wikipedia.org/wiki/Portage), совмещающая в себе возможности конфигурирования, настройки, а также автоматизированную систему управления пакетами, благодаря набору утилит на Python и Bash, призванных облегчить и упорядочить установку программного обеспечения из исходных кодов или бинарных пакетов, с учётом всех зависимостей. Последняя создавалась под влиянием системы управления пакетами в [FreeBSD](https://ru.wikipedia.org/wiki/FreeBSD), называемой [портами](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D1%8B_FreeBSD).
* Аппаратная многоплатформенность — на данный момент Gentoo портирована на [ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)), [x86](https://ru.wikipedia.org/wiki/X86), [x86-64](https://ru.wikipedia.org/wiki/X86-64), [PowerPC](https://ru.wikipedia.org/wiki/PowerPC), [PowerPC 970](https://ru.wikipedia.org/wiki/PowerPC_970) (PowerPC G5), [SPARC](https://ru.wikipedia.org/wiki/SPARC), [MIPS](https://ru.wikipedia.org/wiki/MIPS_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)), [DEC Alpha](https://ru.wikipedia.org/wiki/DEC_Alpha), [PA-RISC](https://ru.wikipedia.org/wiki/PA-RISC), [IBM/390](https://ru.wikipedia.org/wiki/IBM_System/390), [SuperH](https://ru.wikipedia.org/wiki/SuperH) и [68k](https://ru.wikipedia.org/wiki/Motorola_680x0)[[15]](https://ru.wikipedia.org/wiki/Gentoo_Linux#cite_note-15).
* Оптимизация системы под конкретное аппаратное обеспечение и нужды пользователя. Это достигается посредством сборки программ из исходных текстов с использованием так называемых USE-флагов оптимизации и подключением/отключением необходимых модулей. Для удобства все опции могут быть внесены в конфигурационные файлы как для системы в целом (/etc/portage/make.conf), так и для конкретных программ (/etc/portage/package.use). В случае необходимости изменения флагов, например, в случае подключения поддержки системы печати ([cups](https://ru.wikipedia.org/wiki/Common_UNIX_Printing_System)) или потребности в дополнительных функциях ([kerberos](https://ru.wikipedia.org/wiki/Kerberos), pda), они будут учтены при следующем обновлении, и все программы, где используются эти флаги, включая все зависимости, — автоматически пересобраны. Таким образом, любое обновление программ или системы осуществляется очень просто, например, для всей системы в целом обычно используют emerge -avDNu @world, обновление только системных программ emerge -avDNu @system.
* PHP в Gentoo потребляет до 25 % меньше памяти при прочих равных условиях чем в других Linux-дистрибутивах, [согласно исследованию](http://habrahabr.ru/post/161629/).
* Собственная [init](https://ru.wikipedia.org/wiki/Init)-система на базе проекта [OpenRC](https://ru.wikipedia.org/wiki/OpenRC), расширенная и удобная система инициализационных файлов, в частности, вместо числовых используются именованные уровни запуска (runlevels), при этом с указанием зависимости от прочих сценариев. Для управления используется команда rc-update.
* Маски́рование ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *masking*) — возможность использовать как стабильные (по умолчанию), так и экспериментальные, но более свежие функциональные версии программ. При этом сохраняется возможность откатиться на любую из старых версий (также следует учитывать, что есть ряд пакетов и подсистем, не поддерживающих возврат к старым версиям, что связано с особенностями их функционирования, например, библиотека glibc).
* Более 10000 пакетов в основном дереве и множество подключаемых [оверлеев](https://ru.wikipedia.org/wiki/%D0%9E%D0%B2%D0%B5%D1%80%D0%BB%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) от сторонних разработчиков.
* Регулярное обновление пакетов и минимальные сроки устранения уязвимостей. Gentoo позволяет устанавливать несвободные программы через свою основную пакетную систему.

Одним из интересных фактов является то, что в июне 2018 года, коды репозиториев Gentoo в GitHub были взломаны после того, когда злоумышленник получил доступ к учётной записи администратора проекта Gentoo Linux в GitHub, при этом скомпрометировав базу кода. Gentoo оперативно отреагировал на улучшение методов безопасности.

**3. Архитектура ОС**

Gentoo Handbook — это попытка собрать накопленную документацию в связное руководство. Handbook содержит инструкции по установки Gentoo с использованием интернета, а также части о работе с Gentoo и Portage.

Gentoo Linux доступен для множества архитектур. Но что такое архитектура?

Архитектура — это семейство ЦП (центральных процессоров), которые поддерживают одни и те же инструкции. Две самых известных архитектуры в мире десктопов это архитектура x86 и архитектура x86\_64 (которую Gentoo называет amd64). Но существуют и многие другие архитектуры, например, sparc, ppc (семейство PowerPC), mips, arm, и другие.

Gentoo поддерживает множество архитектур. По этой причине вы обнаружите, что Gentoo Handbook доступен для многих поддерживаемых архитектур. Однако это может привести к некоторому недопониманию, ведь не все пользователи знают о различиях. Некоторые знают только тип и название ЦП, который находится у них в системе (например, i686 или Intel Core i7). Ниже вы найдете общий список поддерживаемых архитектур и их аббревиатуры, которое используется в Gentoo. Однако большинство людей, которые не знают архитектуру своей системы, скорее всего, заинтересованы в x86 или amd64.

**Gentoo Handbook**

Список ниже дает поверхностный обзор архитектур, поддерживаемых различными проектами Gentoo Linux.

[**Alpha Handbook**](https://wiki.gentoo.org/wiki/Handbook:Alpha/ru)

Архитектура Alpha является 64-разрядной архитектурой, разработанной компанией Digital Equipment Corporation (DEC). Она по-прежнему используется в некоторых серверах среднего и верхнего уровня, но развитие ее постепенно затухает. Данная архитектура имеет несколько вариантов - ES40, AlphaPC, UP1000, Noname.

[**AMD64 Handbook**](https://wiki.gentoo.org/wiki/Handbook:AMD64/ru)

Это 64-разрядная архитектура, совместимая с х86 архитектурой (и так же известная как x86\_64). Впервые была использована фирмой AMD (под именем AMD64) и фирмой Intel (под именем EM64T) и в настоящее время является наиболее известной для компьютеров средней и высокой производительности. Так же встречается в серверном сегменте. Включая AMD Athlon 64, AMD Opteron, AMD Sempron processors, AMD Phenom, Intel Pentium 4, Intel Core2, Intel Core i3, i5, i7 и некоторые варианты Intel Atom.

**ARM Handbook**

Это 32-разрядная архитектура, наиболее распространенная среди малых и встраиваемых систем. Суб-архитектуры в диапазоне от ARMv1 до ARMv7 (Cortex), очень часто встречается в смартфонах, планшетах, игровых консолях, GPS навигаторах, и так далее. Включая StrongARM, Cortex-M

**ARM64 Handbook**

Это новый 64-разрядная вариант системы ARM для встраиваемых и серверных систем. Основная суб-архитектура, называемая AArch64 (также известная как ARMv8-A), выпускается несколькими производителями. Чипы AArch64 встречаются в различных SoC системах, включая платы разработчиков, смартфонов, планшетов, смарт-телевизоров и т.д. Варианты данной архитектуры: ARM Holdings' Cortex-A53, A57, A72, A73 и Qualcomm's Kryo и Falkor.

[**HPPA Handbook**](https://wiki.gentoo.org/wiki/Handbook:HPPA/ru)

Для архитектуры PA-RISC, называемой HPPA, набор инструкций разработан фирмой Hewlett-Packard и использовался в серверах среднего и верхнего класса приблизительно до 2008 (в последствии HP перешли на Intel Itanium). Включая HP 9000 и PA-8600.

[**IA64 Handbook**](https://wiki.gentoo.org/wiki/Handbook:IA64/ru)

Это 64-разрядная архитектура разработана Intel и используются в их процессорах серии Intel Itanium. Эта архитектура не совместим с x86 или x86\_64 (называемая amd64) и встречается в основном в серии серверов среднего и высокого класса. Включая Intel Itanium.

[**MIPS Handbook**](https://wiki.gentoo.org/wiki/Handbook:MIPS/ru)

Разработанная MIPS Technologies, архитектура MIPS влечет за собой несколько подсемейств (так называемые ревизии), такие как MIPS I, MIPS III, MIPS32, MIPS64 и далее. Она является наиболее распространенной среди встраиваемых систем. Включая MIPS32 1074K, R16000.

[**PPC Handbook**](https://wiki.gentoo.org/wiki/Handbook:PPC/ru)

Эта 32-разрядная архитектура используется многими процессорами Apple, IBM и Motorola. Она наиболее часто встречается во встраиваемых системах. Включая Apple OldWorld, Apple NewWorld, generi Pegasos, Efika, старые IBM iSeries и pSeries.

[**PPC64 Handbook**](https://wiki.gentoo.org/wiki/Handbook:PPC64/ru)

Это 64-разрядный вариант архитектуры PPC, популярный как во встраиваемых системах, так и в производительных серверах высокого класса. Включая IBM RS/6000s, IBM pSeries, IBM iSeries.

[**SPARC Handbook**](https://wiki.gentoo.org/wiki/Handbook:SPARC/ru)

Архитектура SPARC является довольно известной, благодаря ее распространителям Sun (ныне Oracle) и Fujitsu. Она используется не только в серверных системах, но и в нескольких рабочих станциях. В Gentoo, поддерживаются только SPARC64 процессоры. Включая E3000, Blade 1000, Ultra 2.

[**X86 Handbook**](https://wiki.gentoo.org/wiki/Handbook:X86/ru)

Это 32-разрядная архитектура используется процессорами, про которые часто говорят, "Intel совместимые". Это была, до недавнего времени, наиболее популярной архитектурой для настольных ПК. Gentoo предоставляет билды i486 (поддерживается все семейство) и i686 (для процессоров Pentium и выше, или совместимых). Включая i486, i686, AMD Athlon, Intel Core, Intel Atom.

**Немного о самом Linux:**

Linux является многозадачной многопользовательской операционной системой, поддерживаемой большинство свойств, присущих реализациям UNIX, которых больше нигде нет.

Это означает, что одновременно много пользователей могут работать на одной машине, одновременно выполнять много программ.

Это показывает, что эта система хорошо совместима с рядом стандартов для UNIX, на уровне исходных текстов, включая IEEE POSIX.1, System V и BSD. Он создавался, имея в виду такую совместимость. Поэтому, скорее всего, вы найдете в этой ОС черты, присущие многим UNIX-системам. Большинство свободно распространяемых по сети Internet программ для UNIX может быть откомпилировано для LINUX практически без особых изменений. Кроме того, все исходные тексты для Linux, включая ядро, драйверы устройств, библиотеки, пользовательские программы и инструментальные средства распространяются свободно.

Другие специфические внутренние черты Linux включают контроль работ по стандарту POSIX (используемый оболочками, такими как csh и bash), псевдотерминалы (pty), поддержку национальных и стандартных клавиатур динамически загружаемыми драйверами клавиатур и поддерживает виртуальные консоли (virtual consoles), которые позволяют "переключать экраны" на консоли в текстовом режиме.

Linux поддерживает различные типы файловых систем для хранения данных. Некоторые файловые системы, такие как файловая система ext2fs, были созданы специально для него. Поддерживаются также другие типы файловых систем, такие как Minix-1 и Xenix. Реализована также файловая система MS-DOS, позволяющая прямо обращаться к файлам MS-DOS на жестком диске. Поддерживается также файловая система ISO 9660 CD-ROM для работы с дисками CD-ROM.

Linux обеспечивает полный набор протоколов TCP/IP для сетевой работы. Это включает драйверы устройств для многих популярных карт Ethernet, SLIP (Serial Line Internet Protocol, обеспечивающие вам доступ по TCP/IP при последовательном соединении), PLIP (Parallel Line Internet Protocol), PPP (Point-to-Point Protocol), NFS (Network File System), и так далее. Поддерживается весь спектр клиентов и услуг TCP/IP, таких как FTP, telnet, NNTP и SMTP.

Ядро может само эмулировать команды 387-FPU, так что системы без сопроцессора могут выполнять программы, на него рассчитывающие (т.е. с плавающей точкой). Оно сразу создано с учетом специального защищенного режима для процессоров Intel 80386 и 80486. В частности, Linux использует парадигму описания памяти в защищенном режиме и другие новые свойства процессоров. Любой знакомый с защищенным режимом процессора 80386 знает, что этот чип проектировался для многозадачных систем вроде UNIX (или Mulics).

Само ядро поддерживает загрузку только нужных страниц. То есть с диска в память загружаются те сегменты программы, которые действительно используются. Возможно использование одной страницы, физически один раз загруженной в память, несколькими выполняемыми программами.

Для увеличения объема доступной памяти Linux осуществляет также разбиение диска на страницы: то есть на диске может быть выделено до 256 Мбайт "пространства для свопинга" (swap space). (Swap space не совсем подходящее имя, в Linux в область своппинга выгружается не весь процесс, а только отдельные его части, в которых нет необходимости). Когда системе нужно больше физической памяти, то она с помощью свопинга выводит неактивные страницы на диск. Это позволяет выполнять более объемные программы и обслуживать одновременно больше пользователей. Однако свопинг не исключает наращивания физической памяти, поскольку он снижает быстродействие, увеличивает время доступа.

Ядро также поддерживает универсальный пул памяти для пользовательских программ и дискового кэша. При этом для кэша может использоваться вся память, и наоборот, кэш уменьшается при работе больших программ.

Выполняемые программы используют динамически связываемые библиотеки, т.е. выполняемые программы могут совместно использовать библиотечную программу, представленную одним физическим файлом на диске (иначе, чем это реализовано в механизме разделяемых библиотек SunOS). Это позволяет выполняемым файлам занимать меньше места на диске, особенно тем, которые многократно используют библиотечные функции. Есть также статические связываемые библиотеки для тех, кто желает пользоваться отладкой на уровне объектных кодов или иметь "полные" выполняемые программы, которые не нуждаются в разделяемых библиотеках. В Linux разделяемые библиотеки динамически связываются во время выполнения, позволяя программисту заменять библиотечные модули своими собственными.

Для обеспечения отладки ядро Linux выдает дампы памяти для "посмертного" анализа. Использование дампа и динамических отладчиков позволяет определить причины краха программы.

**4. Установка дистрибутива Linux Gentoo**

**Все, конечно, просто… но не совсем.**

Несмотря на бодрое начало, надо уточнить, что ряд сложностей в сравнении с другими (например, дебианом и пресловутой убунтой) все же присутствует.

1. Без документации никуда. Вам придется немало прочесть, чтобы установить этот дистрибутив. Крайне желательно, чтобы она была прочитана на английском, т.к. русская версия не всегда имеет актуальную версию
2. Консоль — наше все. Если у вас консолефобия, вам придется побороть себя. Она тут всегда и везде: установка, обновление, настройка, все это и многое другое.
3. Если вам срочно понадобится переустановить систему, или поставить её на новую машину, то это может занять довольно много времени. Помните об этом, когда ставите gentoo в продакшн.

**Приложения и стабильность**

Ну, со спорными моментами, вроде, закончили (почти), а теперь вкусности! Я не просто так объединил эти два пункта. Именно метод установки, обновления и удаления приложений вносит существенный вклад в стабильность системы. Приложений много. Очень много.

* Установка всего этого многообразия происходит через утилиты *emerge* и *eix*. Замечательной чертой gentoo является действительно стабильная «стабильная ветка», совмещенная с возможностью безболезненной установки свежих версий нужных вам программ. В принципе, можно полностью сидеть на нестабильном ПО, просто зачем? И наоборот, зная, что какие-то версии плохо работают в моем случае, есть возможность не обновляться на них.
* В разрешение проблем с зависимостями вносят свой вклад система слотов, дающая возможность устанавливать множество версий одной библиотеки, и отключение зависимостей через отключение USE-флагов. Часто пугают необходимостью собирать ядро, но на самом деле в этом нет ничего сложного, и с помощью *genkernel* не требует какого-то особого мастерства.
* Те, кто заботится о карме лицензионной чистоте, могут указать с какими лицензиями можно ставить ПО.
* Rolling-релизы делают обновление более безболезненным.
* Утилита eselect поможет выставить ряд переменных среды, например, дефолтные питон и яву.

**4.1. Подготовка дисков**

Для начала, рассмотрим блочные устройства. Наиболее известным блочным устройством можно считать первый диск в системе Linux, именуемый /dev/sda. И SCSI-, и Serial ATA-диски обозначаются как /dev/sd\*; благодаря фреймворку ядра libata даже IDE-диски обозначаются как /dev/sd\*. Если же используется старый фреймворк устройств, первым IDE-диском будет /dev/hda.

Вышеназванные блочные устройства представляют абстрактный интерфейс к диску. Пользовательские приложения могут использовать их для взаимодействия с диском, не заботясь о том, какой это диск — IDE, SCSI или ещё какой-либо. Программа просто адресует пространство на диске как совокупность следующих друг за другом 512-байтных блоков с произвольным доступом.

**Таблица разделов**

Несмотря на то, что теоретически возможно использовать весь (raw) не разбитый диск для размещения системы Linux (например, при создании btrfs RAID), этого почти никогда не случается на практике. Вместо этого, блочные устройства (диск) разбивается на меньшие, более удобные для обращения, блочные устройства. В системах **amd64**, они называются разделами. В настоящее время есть две стандартных технологии разметки дисков: MBR и GPT.

**MBR**

*MBR (Master Boot Record, главная загрузочная запись)* использует 32-битные идентификаторы для определения начала сектора и длины раздела, поддерживает три типа разделов: основные, расширенные и логические. Информация о расположении основных разделов хранится в самой главной загрузочной записи - очень небольшом (обычно 512 байт) месте в самом начале диска. В связи с этим небольшим размером поддерживаются только четыре основных раздела (например, от /dev/sda1 до /dev/sda4).

Каждый раздел имеет ограничение размера в 2 ТБ (в связи с 32-битными идентификаторами). Кроме того, MBR не обеспечивает резервного копирования главной загрузочной записи, так что если пользователь или приложение перезапишет MBR, то вся информация о разделах теряется.

**GPT**

*GPT (GUID Partition Table, таблица разделов GUID)* использует 64-битные идентификаторы разделов. Место, в котором храниться информация о разделах также гораздо больше, чем 512 байт в MBR. Это подразумевает что почти нет никаких ограничений на количество разделов в разбитом с помощью GPT диске. Также предельный размер раздела гораздо больше (почти 8 ЗБ - да, зеттабайт).

Когда программным интерфейсом системы между операционной системой и прошивкой является UEFI (вместо BIOS), GPT является почти обязательным, так как с MBR будут возникать проблемы совместимости.

Также GPT использует контрольные суммы и избыточность. Он содержит контрольные суммы CRC32 для обнаружения ошибок в заголовке и таблице разделов. У GPT есть резервная таблица в конце диска. Ее можно использовать для восстановления первичной таблицы GPT, которая располагается в начале диска.

**Использование parted для создания разделов диска**

Пример разметки разделов, упомянутый ранее в инструкции, будет использоваться такой:

|  |  |
| --- | --- |
| **Раздел** | **Описание** |
| /dev/sda1 | BIOS boot раздел |
| /dev/sda2 | Boot раздел |
| /dev/sda3 | Swap раздел |
| /dev/sda4 | Root раздел |

Приложение **parted** предлагает простой интерфейс для разбития дисков и поддерживает очень большие разделы (более 2 ТБ). Указывайте для **parted** диск (например, в нашем примере мы используем /dev/sda). Рекомендуется также для **parted** указывать оптимальное выравнивание разделов:

**root #**parted -a optimal /dev/sda

Выравнивание разделов означает, что разделы будут начинаться с известных границ, в пределах диска, гарантируя, что операции на диске от уровня операционной системы (извлечения страниц с диска) будут использовать наименьшее количество внутренних операций диска. Невыровненные разделы могут потребовать извлечения двух страниц, даже если операционная система запросила только одну.

Чтобы узнать обо всех опциях, поддерживаемых parted введите **help**.

**Установка метки GPT**

Большинство дисков в архитектурах **x86** или **amd64** создаются с меткой *msdos*. Используйте **parted** для размещения метки *GPT* на диске используйте команду **mklabel gpt**:

**Предупреждение**  
Изменение типа разделов удалит все разделы на диске. Вся информация на диске будет потеряна.

**(parted)**mklabel gpt

Чтобы сделать структуру разделов MBR используйте команду **mklabel msdos**.

**Создание разделов**

Теперь, с помощью **parted**, создадим разделы со следующими параметрами:

* Используемый тип раздела. Обычно это *основной* раздел. Если используется для разделов метка msdos, то учтите, что основных разделов может быть только 4. Если необходимо больше чем 4 раздела, то необходимо сначала создать *расширенный* раздел, а затем создать внутри него *логические*.
* Начальная позиция раздела (которая может быть выражена в MB, GB, ...)
* Конечная позиция раздела (которая может быть выражена в MB, GB, ...)

Во-первых, необходимо сказать *parted*, что единицы, с которыми мы работаем мегабайты (на самом деле мебибайты, сокращенно MiB, являющиеся "стандартом" обозначения, но мы будем использовать MB в тексте, поскольку это намного более распространено):

**(parted)**unit mib

Теперь создайте 2МБ раздел, который будет использоваться загрузчиком GRUB2 позже. Используйте команду **mkpart** и сообщите **parted**, что раздел начинается с 1 МБ от начала диска и заканчивается на 3 МБ (для создания раздела размером 2 МБ).

**(parted)**mkpart primary 1 3

**(parted)**name 1 grub

**(parted)**set 1 bios\_grub on

Сделайте тоже самое для загрузочного раздела (128 Мб), раздела подкачки (в примере 512 МБ) и корневого раздела, который занимает все оставшееся пространство (для него конец помечается как -1, что означает конец диска минус один МБ, это подходит для последнего раздела).

**(parted)**mkpart primary 3 131

**(parted)**name 2 boot

**(parted)**mkpart primary 131 643

**(parted)**name 3 swap

**(parted)**mkpart primary 643 -1

**(parted)**name 4 rootfs

При использовании интерфейса UEFI для загрузки системы (вместо BIOS), пометьте загрузочный раздел как системный раздел EFI. Parted делает это автоматически, когда устанавливается опция *boot* у раздела:

**(parted)**set 2 boot on

**Заметка**  
При установке на UEFI, на разделе boot будут отображаться флаги boot и esp (efi system partition).

Используйте команду **quit** для выхода из parted.

Далее, используя **mkfs.fat**, создайте файловую систему FAT в системном разделе EFI и добавьте его в /etc/fstab как в примере ниже:

**root #**mkfs.fat -F 32 -n efi-boot /dev/sda1

**root #**mkdir /boot/efi

**Файл** **/etc/fstabДобавление записи для /boot/efi**

/dev/sda1 /boot/efi vfat noauto,noatime 1 2

**root #**mkfs.ext4 /dev/sda4

Теперь созданы файловые системы на вновь созданных томах (или логических разделах).

**Активация раздела подкачки**

Для инициализации разделов подкачки используется команда **mkswap**:

**root #**mkswap /dev/sda3

Чтобы активировать раздел подкачки, используйте **swapon**:

**root #**swapon /dev/sda3

**Монтирование корневого раздела**

Теперь, когда созданы разделы и файловые системы на них, настало время их смонтировать. Используйте команду **mount**, но не забывайте, что необходимо создать каталоги для монтирования каждого созданного раздела. В качестве примера мы смонтируем корневой раздела:

**root #**mount /dev/sda4 /mnt/gentoo

**4.2. Установка установочных файлов Gentoo**

**Установка времени и даты, Ручная настройка**

Можно воспользоваться командой **date**, чтобы вручную настроить системные часы. Используйте синтаксис MMDDhhmmYYYY ((M)Месяц, (D)День, (h)час, (m)минута и (Y)Год).

Например, чтобы установить дату на 13:16 3 октября 2016 года:

**root #**date 100313162016

**Выбор stage архива**

Выбор базового архив для системы может сэкономить значительное количество времени, в дальнейшем в процессе установки, в частности, во время выбора системного профиля. Выбор архива будет непосредственно влиять на будущую конфигурацию системы и может сохранить от головной боли или двух последующих.

**Загрузка архива stage**

Когда корневая файловая система подключена, перейдите в точку монтирования Gentoo (скорей всего это /mnt/gentoo):

**root #**cd /mnt/gentoo

Чтобы загрузить файл архива stage, просмотрите список зеркал Gentoo так:

**root #**links https://www.gentoo.org/downloads/mirrors/

В списке зеркал выберите зеркало которое находится рядом. Обычно зеркала с HTTP протоколом хватает, но и другие протоколы также доступны. Перейдите в каталог releases/amd64/autobuilds/. Там отображаются все доступные stage файлы (так же они могут быть сохранены в подкаталогах с названиями отдельных суб-архитектур). Выберите нужный и нажмите d для загрузки.

**Распаковка архива stage**

Теперь распакуем загруженный stage архив в систему. Далее воспользуемся **tar**:

**root #**tar xvpf stage3-\*.tar.bz2 --xattrs --numeric-owner

Убедитесь, что оба параметра (**xvpf** и --**xattrs**) используются. Позиция **x** указывает на Извлечение, **v** на Подробные Сообщения, чтобы видеть то, что происходит во время процесса извлечения (опционально) **p** для Сохранения прав доступа и **f** для обозначения, что мы хотим извлечь файл, а не стандартный ввод. –**xattrs** позволит также включить расширенные атрибуты, хранящиеся в архиве. Наконец, --**numeric-owne** используется, для того чтобы убедиться, что идентификаторы пользователей и групп, в распакованных файлах из архива, останутся такими же, как и задумывалось командой Gentoo по подготовке релизов (release engineering), даже если предприимчивые пользователи не используют официальный установочный носитель Gentoo.

**Настройка параметров компиляции**

Для оптимизации Gentoo можно установить пару переменных, которые повлияют на поведение Portage, официально поддерживаемый пакетный менеджер в Gentoo. Все переменные могут быть установлены как переменные среды (с помощью **export**), но это не является постоянным. Чтобы использовать сохраненные настройки Portage читает файл /etc/portage/make.conf; конфигурационный файл для Portage.

Запустите редактор (в этом руководстве мы используем **nano**) для изменения параметров оптимизации о которых написано далее.

**root #**nano -w /mnt/gentoo/etc/portage/make.conf

**CFLAGS и CXXFLAGS**

*CFLAGS* и *CXXFLAGS* переменные определяют параметры оптимизации для GCC C и C++ компиляторов соответственно. Хотя они и указаны здесь, для достижения максимальной производительности можно было бы указать флаги оптимизации для каждой программы отдельно. Причина этого в том, что все программы различны. Но этим тяжело управлять, следовательно, запишем эти переменные в make.conf файл.

В make.conf следует указывать параметры оптимизации, которые сделают систему наиболее отзывчивой в целом. Не нужно использовать экспериментальные настройки; излишняя оптимизация может привести к непредсказуемому поведению программ (аварийный выход из программы, или еще хуже, к неправильной работе).

Определение переменных *CFLAGS* и *CXXFLAGS* позволяет комбинировать несколько флагов оптимизации в одной строке. Значения по умолчанию, содержащиеся в архив stage3, наиболее хороши. Ниже один пример:

**Код** **Пример для переменных CFLAGS и CXXFLAGS**

CFLAGS="-march=native -O2 -pipe"

*# Используйте те же настройки для обеих переменных*

CXXFLAGS="**${**CFLAGS**}**"

**MAKEOPTS**

*MAKEOPTS* переменная определяет, сколько параллельных процессов компиляции должно запускаться при установке пакета. Хорошим вариантом будет использовать количество процессоров (или ядер процессора) в системе плюс один, но это вариант не всегда идеален.

**Код** **Пример записи MAKEOPTS в make.conf**

MAKEOPTS="-j2"

Обновите /mnt/gentoo/etc/portage/make.conf файл в соответствии с личными предпочтениями и сохраните изменения (пользователям nano нужно нажать Ctrl+X).

**4.3. Установка базовой системы Gentoo**

**Chrooting**

Для быстрой загрузки исходного кода рекомендуется выбрать быстрое зеркало. Portage будет искать в файле make.conf переменную *GENTOO\_MIRRORS* и использовать перечисленные в ней зеркала. Можно просмотреть список зеркал Gentoo и найти зеркало (или зеркала), наиболее близко расположенное к месту физического расположения (они чаще всего и есть самые быстрые). Тем не менее, мы предоставляем хороший инструмент под названием **mirrorselect**, который предлагает удобный интерфейс для выбора подходящего зеркала. Просто перейдите на нужное зеркало и нажмите пробел для выбора одного или нескольких.

**root #**mirrorselect -i -o >> /mnt/gentoo/etc/portage/make.conf

**Gentoo репозиторий ebuild-файлов**

Вторым важным шагом в выборе зеркала является настройка Gentoo репозитория ebuild-файлов в файле /etc/portage/repos.conf/gentoo.conf. Этот файл содержит информацию, необходимую для обновления пакетного репозитория (коллекции ebuild и связанных с ними файлов, содержащих всю информацию, необходимую Portage для загрузки и установки пакетов программного обеспечения).

Настройку репозитория можно сделать весьма просто. Сперва, создайте [repos.conf](https://wiki.gentoo.org/wiki/etc/portage/repos.conf/ru) каталог, если он не существует:

**root #**mkdir /mnt/gentoo/etc/portage/repos.conf

Далее, скопируйте файл конфигурации Gentoo репозитория, предоставляемый Portage, в (только что созданный) каталог repos.conf:

**root#**cp/mnt/gentoo/usr/share/portage/config/repos.conf/mnt/gentoo/etc/portage/repos.conf/gentoo.conf

**Копирование информации о DNS**

Единственное, что еще осталось сделать перед входом в новое окружение, это скопировать информацию о DNS из файла /etc/resolv.conf. Это нужно сделать, чтобы гарантировать, что сеть все еще будет работать даже после входа в новое окружение. Файл /etc/resolv.conf содержит сервера имен (DNS).

**root #**cp -L /etc/resolv.conf /mnt/gentoo/etc/

**Подключение необходимых файловых систем**

Скоро Linux root будет перемещен в новое место. Чтобы новое окружение работало должным образом, для него должны быть доступны некоторые файловые системы.

Файловые системы, которые должны быть доступны:

* /proc/ - это псевдо-файловая система (она выглядит как обычные файлы, но на самом деле генерируется на лету) из которой ядро Linux предоставляет информацию для окружения
* /sys/ - это псевдо-файловая система, как и /proc/, которую она однажды заменит, также она более структурирована, чем /proc/
* /dev/ - это обычная файловая система, частично управляемая менеджером устройств Linux (обычно **udev**), которая содержит все файлы устройств

Каталог /proc/ монтируется в /mnt/gentoo/proc/, а остальные два каталога через bind-mounted. Дальнейшее означает, что, например, /mnt/gentoo/sys/ на самом деле *будет* /sys/ (это просто вторая точка входа в ту же файловую систему), тогда как /mnt/gentoo/proc/ является новой точкой монтирования (так сказать, экземпляром) файловой системы.

**root #**mount -t proc /proc /mnt/gentoo/proc

**root #**mount --rbind /sys /mnt/gentoo/sys

**root #**mount --make-rslave /mnt/gentoo/sys

**root #**mount --rbind /dev /mnt/gentoo/dev

**root #**mount --make-rslave /mnt/gentoo/dev

**Переход в новое окружение**

Теперь, когда все разделы инициализированы и базовое окружение установлено, нашстало время войти в новое установочное окружение, перенеся корневой каталог в него. Это означает, что сессия изменит свой корень (наивысший каталог, который может быть доступен) из текущей установочного окружения (CD или другого установочного носителя) в систему установки (т.е размеченных разделов). Отсюда и название, сменить корень (change root - chroot).

Chroot делается в три шага:

1. Изменение корневой директории с / (который находится на установочном носителе) в /mnt/gentoo/ (на разделах диска) используя chroot
2. С помощью команды **source**, некоторые параметры (перечисленные в /etc/profile) загружаются в память
3. Изменение приглашения командной строки, чтобы запомнить, что эта сессия находится в chroot окружении.

**root #**chroot /mnt/gentoo /bin/bash

**root #**source /etc/profile

**root #**export PS1="(chroot) $PS1"

**Монтирование раздела boot**

mount /boot/efi

**root #**mount /dev/sda2 /boot

**Настройка Portage**

Следующим шагом будет установка снимка основного репозитория ebuild-файлов. Этот снимок содержит коллекцию файлов, которая сообщает Portage о доступных программах (для установки), какой профиль может выбрать системный администратор, о новостях о конкретных пакетах или профилях и так далее.

**emerge-webrsync** рекомендуется использовать тем, кто находится за фаерволом (для загрузки снимка используется только протоколы HTTP/HTTPS), а также снижает нагрузку на канал сети. У кого нет ограничений с сетью или шириной канала, могут с радостью перейти к следующему разделу.

Команда ниже загрузит последний снимок (которые выпускаются каждый день), с одного из зеркал Gentoo, и распакует его в системе:

**root #**emerge-webrsync

**Обновление Gentoo репозитория ebuild-файлов**

**root #**emerge –sync

**Выбор подходящего профиля**

*Профиль* - это важная часть любой системы Gentoo. Он не только определяет переменные *USE*, *CFLAGS*, и другие важные переменные, а также заставляет систему использовать только определенные версии пакетов. Все эти нюансы поддерживаются разработчиками Portage в Gentoo.

**root #**eselect profile set 12

**Обновление @world**

Теперь, когда выбран новый системный профиль, то разумно будет обновить [@world](https://wiki.gentoo.org/wiki/World_set_(Portage)) чтобы базовая часть системы изменилась с учетом нового профиля.

Это действие *необходимо* делать тем, кто выбрал профиль с systemd (так как все Gentoo официальные stage архивы используют по умолчанию OpenRC как систему инициализации), однако это *необязательно* для других профилей:

**root #**emerge –ask --update --deep --newuse @world

**Настройка USE переменной**

*USE* - это одна из самых мощных переменных Gentoo доступная пользователям. Разные программы могут быть скомпилированы с или без поддержки некоторых опций. Например, некоторые программы могут быть скомпилированы с поддержкой GTK+ или поддержкой Qt. Другие могут быть скомпилированы с или без поддержки SSL.

**/etc/portage/make.conf**

**Настройки USE для системы ориентированной для использования KDE с поддержкой DVD, ALSA и записи CD**

USE="-gtk -gnome qt4 qt5 kde dvd alsa cdr"

**Часовой пояс**

Определите какой часовой пояс нужен для системы. Посмотреть доступные часовые пояса можно в /usr/share/zoneinfo/, затем запишите его в файл /etc/timezone.

**root #**ls /usr/share/zoneinfo

**root #**echo "Europe/ Ulyanovsk " > /etc/timezone

Далее, перенастроим пакет sys-libs/timezone-data, что обновит /etc/localtime файл, основываясь на записи в /etc/timezone. Файл /etc/localtime используется системной библиотекой C, чтобы узнать в каком часовом поясе находится система.

**root #**emerge --config sys-libs/timezone-data

**Настройка локалей**

Большинству пользователей достаточно иметь одну или две локали на своих системах.

Локаль, на самом деле, указывает не только язык, который использует пользователь при взаимодействии с системой, но и правила для сортировки строк, формат вывода даты и времени, и так далее.

Локали, которые должна поддерживать система, должны быть перечислены в /etc/locale.gen.

**root #**nano -w /etc/locale.gen

Следующие локали являются примером для получения английской (США) и немецкой (Germany) локалей с поддержкой формата символов (например, UTF-8).

Далее, запустим **locale-gen**. Это сгенерирует локали, которые были перечислены в файле /etc/locale.gen.

**root #**locale-gen

Команда **eselect locale set VALUE** может установить нужную локаль:

**root #**eselect locale set 3

Заново перезагрузите окружение:

**root #**env-update && source /etc/profile && export PS1="(chroot) $PS1"

**4.4. Настройка ядра Linux**

**Установка исходного кода**

Все дистрибутивы строятся вокруг ядра Linux. Ядро является прослойкой между пользовательским программным обеспечением и физическим оборудованием. Gentoo предоставляет несколько вариантов исходного кода ядра.

Для систем, основанных на amd64 архитектуре, рекомендуется пакет sys-kernel/gentoo-sources.

Выберем подходящий исходный код ядра и установим с помощью **emerge**:

**root #**emerge --ask sys-kernel/gentoo-sources

Данная команда установит исходный код ядра Linux в /usr/src/, в котором символьная ссылка linux будет указывать на установленную версию:

**root #**ls -l /usr/src/linux

lrwxrwxrwx 1 root root 12 Oct 13 11:04 /usr/src/linux -> linux-3.16.5-gentoo

Теперь следует настроить и собрать ядро. Здесь существует два основных подхода:

1. Ядро настраивается и собирается вручную.
2. Утилита **genkernel** собирает и устанавливает его автоматически.

В данной инструкции мы расскажем, как провести настройку вручную, поскольку это лучший способ оптимизировать окружение.

**Genkernel**

Приступим. Сперва, нужно установить sys-kernel/genkernel:

**root #**emerge --ask sys-kernel/genkernel

Затем отредактируйте файл /etc/fstab, где следует указать в строке /boot/ правильное устройство во втором поле. Если следовали примеру из данной инструкции, то, скорее всего, это устройство /dev/sda2 с файловой системой ext2 и строка должна выглядеть следующим образом:

**root #**nano -w /etc/fstab

**/etc/fstabПример файла /etc/fstab с UEFI и swap-разделом:**

/dev/sda1 /boot/efi vfat noauto,noatime 1 2

/dev/sda2 none swap sw 0 0

/dev/sda3 / ext4 noatime 0 1

Осталось скомпилировать ядро, выполнив **genkernel all**. Учтите, что поскольку **genkernel** включает поддержку как можно большего диапазона оборудования, процесс сборки может занять некоторое время!

**root #**genkernel all

В некоторых случаях необходимо включить поддержку initramfs - an initial ram-based file system (файловая система, основанная на оперативной памяти). Наиболее распространённая причина — это нахождение различных системных путей (вроде /usr/ или /var/) на различных разделах. При наличии initramfs эти разделы могут быть подгружены средствами initramfs.

Без initramfs существует вероятность неправильной загрузки, так как для запуска некоторых ФС может потребоваться информация на этих самых разделах.

Для установки initramfs, сперва нужен sys-kernel/genkernel, который его сгенерирует:

**root #**emerge --ask sys-kernel/genkernel

**root #**genkernel --install initramfs

**Установка файлов прошивки**

Для корректной работы некоторых драйверов требуется установка дополнительных файлов прошивки. Часто подобное требуется для сетевых интерфейсов, особенно беспроводных. Большинство файлов прошивки поставляется в sys-kernel/linux-firmware:

**root #**emerge --ask sys-kernel/linux-firmware

**4.5. Настройка параметров системы**

**Настройка сети**

Во время установки Gentoo Linux сеть была уже настроена. Но она была настроена для установочного CD, а не для установленной среды. Сейчас настроим сеть для установленной системы Gentoo Linux.

Сначала, установите net-misc/netifrc:

**root #**emerge –ask --noreplace net-misc/netifrc

Если сетевое соединение требует настройки в связи с определенными опциями DHCP или потому что использования DHCP не достаточно, тогда откройте /etc/conf.d/net:

**root #**nano -w /etc/conf.d/net

**Файл** **/etc/conf.d/netНастройка статического IP-адреса**

config\_eth0="192.168.0.2 netmask 255.255.255.0 brd 192.168.0.255"

routes\_eth0="default via 192.168.0.1"

Для того, чтобы сетевые интерфейсы начинали работать во время загрузки системы их нужно добавить к уровню запуска по умолчанию.

**root #**cd /etc/init.d

**root #**ln -s net.lo net.eth0

**root #**rc-update add net.eth0 default

**Файл hosts**

Далее сообщим Linux о сетевом окружении.

**root #**nano -w /etc/hosts

**Файл** **/etc/hostsВнесение сетевой информации**

# Это настройки для текущей системы и должно быть настроено

127.0.0.1 tux.homenetwork tux localhost

**Системная информация Root пароль**

Изменить пароль root можно с помощью команды **passwd**.

**root #**passwd

**4.6. Установка системных средств**

**Системный журнал**

Некоторые утилиты не были включены в stage3, так как некоторые пакеты обеспечивают схожую функциональность. Теперь пользователь должен установить те, которые ему требуются.

Для инсталляции выбранной программы системного журнала, установите ее с помощью команды emerge и добавьте в уровень запуска по умолчанию с помощью **rc-update**. Следующий пример покажет как это сделать для app-admin/sysklogd:

**root #**emerge --ask app-admin/sysklogd

**root #**rc-update add sysklogd default

**Cron демон**

Cron демон выполняет команды по расписанию. Это очень удобно, если некоторые команды требуется выполнять регулярно (например, ежедневно, еженедельно или ежемесячно).

**root #**emerge --ask sys-process/cronie

**root #**rc-update add cronie default

**Индексирование файлов**

Для того чтобы иметь возможность индексировать файловою систему, которая обеспечит более быстрый доступ к файлам, установите sys-apps/mlocate.

**root #**emerge --ask sys-apps/mlocate

**Утилиты для файловых систем**

В зависимости от используемых файловых систем, необходимо установить необходимые утилиты для обслуживания их (для проверки целостности файловых систем, создания дополнительных файловых систем, и т.п.). Обратите внимание, что инструмент для управления файловыми системами ext2, ext3 и ext4 (sys-fs/e2fsprogs) уже установлен как часть @system.

В следующей таблице перечислены утилиты, которые требуется установить, если используется определенная файловая система:

|  |  |
| --- | --- |
| **Filesystem** | **Package** |
| Ext2, 3, и 4 | sys-fs/e2fsprogs |
| XFS | sys-fs/xfsprogs |
| ReiserFS | sys-fs/reiserfsprogs |
| JFS | sys-fs/jfsutils |
| VFAT (FAT32, ...) | sys-fs/dosfstools |
| Btrfs | sys-fs/btrfs-progs |

**Сетевые утилиты**

Для того чтобы система автоматически получила IP адрес для одного или более сетевого интерфейса, когда запускается netifrc скрипт, необходимо установить клиент DHCP. Мы рекомендуем использовать net-misc/dhcpcd хотя и много других DHCP клиентов доступны в Gentoo репозитории:

**root #**emerge --ask net-misc/dhcpcd

**4.7. Настройка начального загрузчика**

**Выбор загрузчика**

Когда ядро Linux настроено, системные утилиты установлены и конфигурационные файлы отредактированы, настало время для установки последней важной части Linux-системы: загрузчика.

Загрузчик отвечает за загрузку ядра Linux во время загрузки - без него система не будет знать, как действовать, если нажата кнопка питания.

Для **amd64**, мы написали как настроить либо GRUB2, либо LILO для систем на базе BIOS, и GRUB2 или efibootmgr для UEFI систем.

**GRUB2**

**root #**emerge –ask system

Если используется старая материнская плата, BIOS которой поддерживает только таблицу разделов MBR, никаких дополнительных настроек не требуется для того, чтобы установить GRUB:

**root #**emerge –ask --verbose sys-boot/grub:2

**Установка GRUB2 на диск**

Далее установим необходимые для Grub2 файлы в каталог /boot/grub/ с помощью команды **grub-install**. Если предположить, что первый диском (тот, с которого система загружается) является /dev/sda2, то одна из следующих команд сделает это:

* Когда используется BIOS:

**root #**grub-install /dev/sda2

**Настройка**

Далее, нужно сгенерировать конфигурационный файл GRUB2 на основе настроек пользователя, указанных в файле /etc/default/grub и в скриптах /etc/grub.d. В большинстве случаев, ничего не нужно настраивать пользователям, так как GRUB2 автоматически определяет, какое ядро есть для загрузки (самый высокий приоритет у /boot/) и какая файловая система у rootfs. Здесь также можно добавить параметры ядра в /etc/default/grub, используя переменную *GRUB\_CMDLINE\_LINUX*.

Для создания окончательной конфигурации GRUB2, запустите команду **grub-mkconfig**:

**root #**grub-mkconfig -o /boot/grub/grub.cfg

**Перезагрузка системы**

Выйдите из chroot среды и размонтируйте все смонтированные разделы. Затем введите ту самую волшебную команду, которая запускает последний настоящий тест: **reboot**.

**root #**exit

**cdimage ~#**cd

**cdimage ~#**umount -l /mnt/gentoo/dev{/shm,/pts,}

**cdimage ~#**umount -R /mnt/gentoo

**cdimage ~#**reboot

**4.8. Завершение установки**

**Управление учетными записями**

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

**Login:**root

Password: (Enter the root password)

**root #**useradd -m -G users,wheel,audio -s /bin/bash larry

**root #**passwd larry

**Очистка диска, удаляем tar-архивы**

Теперь, когда установка Gentoo закончена и система была перезагружена, если все прошло хорошо, теперь можно удалить скачанный архив tar со stage3 с жесткого диска. Помните, что он был скачан в каталог /.

**root #**rm /stage3-\*.tar.bz2\*

**5. Установка графической оболочки Xfce**

Перед начало установки графической оболочки, стоял вопрос: "Какую все же ставить? ". На первый план выдвинулась гр. оболочка под названием "Cinnamon" - это современное окружение рабочего стола с традиционным интерфейсом, как у GNOME 2, LXDE или Xfce, что вполне может быть удобным для работы. Cinnamon хорошо выглядит и имеет много функциональных возможностей. Cinnamon - это форк Gnome Shell и разработаны для Linux Mint. Он доступен для архитектур amd64 и x86.

Все конечно здорово, кроме одного фичи. Когда предстояла установка Cinnamon, используя команду в терминале:

emerge --ask gnome-extra/cinnamon

Появилась ошибка: где Webkit-gtk не смог установиться, а соответственно и зависящие от него пакеты, в том числе и Cinnamon.

На замену пришла Xfce - это легковесное окружение рабочего стола для UNIX-подобных операционных систем. Xfce старается быть быстрым и не предъявляет больших требований к системным ресурсам, оставаясь при этом визуально привлекательным и удобным для пользователей."

**Установка Emerge**

Пришло время установить рабочий стол Xfce!

Без явной установки xfce-extra/xfce4-notifyd с помощью команды emerge, пакет virtual/notification-daemon установит зависимый пакет из среды GNOME x11-misc/notification-daemon. Таким образом, мы должны установить xfce-base/xfce4-meta вместе с xfce-extra/xfce4-notifyd, и затем исключить последний из файла world:

root #emerge --ask xfce-base/xfce4-meta xfce-extra/xfce4-notifyd

root #emerge --ask --deselect=y xfce-extra/xfce4-notifyd

Конечно, вместо meta-пакета можно устанавливать только необходимые компоненты, например:

root #emerge --ask xfce-base/xfwm4 xfce-base/xfce4-panel

**Конфигурация Дисплей менеджер**

Для запуска Xfce можно использовать дисплей менеджер. Пожалуйста, обратитесь к статье display manager для его конфигурации.

Большинство дисплей менеджеров используют файлы .desktop для настройки доступных сессий. Пример такого файла:

Файл xfce4.desktop.desktop файл Xfce4 для менеджеров дисплея

Encoding=UTF-8

Name=Xfce4

Comment=Use this session to run Xfce 4 as desktop environment

Exec=/usr/bin/startxfce4

Icon=/usr/share/pixmaps/xfce4\_xicon1.png

Type=Application

**Запуск Xfce без менеджера дисплея**

startx и startxfce4 два доступных варианта для запуска Xfce без использования менеджера дисплея.

startx

Для startx создайте файл ~/.xinitrc и запишите в него:

Файл ~/.xinitrcЗапуск xfce4 при запуске startx exec startxfce4

В случае возникновения проблем с авторизацией или правами доступа в xfce4 при использовании профиля OpenRC (симптомы включают в себя невозможность запуска менеджера питания и невозможность перехода в ждущий или спящий режим), убедитесь, что пакет sys-auth/consolekit установлен с флагом pm-utils. Затем замените вышеуказанную строку в файле ~/.xinitrc на следующую

Файл ~/.xinitrc Правильный способ запуска xfce4 когда вызывается startx

ck-launch-session dbus-launch --sh-syntax --exit-with-session xfce4-session

startxfce4

Для запуска рабочего стола Xfce4 просто наберите startxfce4 в командной строке и нажмите enter:

user $startxfce4

**Дополнительные приложения**

Здесь приведены список дополнительных приложений, являющиеся частью Xfce.

|  |  |
| --- | --- |
| **Пакет** | **Описание** |
| **x11-terms/xfce4-terminal** | Популярные эмулятор терминала для Xfce; замена стандартному **xterm**. |
| **xfce-base/xfce4-appfinder** | Поиск приложений. |
| **xfce-base/thunar** | Файловый менеджер Xfce. |
| **xfce-extra/xfce4-mixer** | Регулирует громкость. |
| **xfce-extra/thunar-volman** | Управляет переносными носителями и дисками. |
| **xfce-extra/tumbler** | Предпросмотр файлов в Thunar. |
| **xfce-extra/thunar-archive-plugin** | Плагин для Thunar, позволяющий работать с архивами; использует app-arch/file-roller. |
| **xfce-extra/xfce4-battery-plugin** | Отображает процент заряда батареи, оставшегося времени, источника питания (AC или от батареи), состояние вентиляторов, предупреждения и возможность настройки запуска команд при определенном уровне питания, которые могут использоваться для перевода ноутбука в гибернацию, если батарея сильно разряжена. |
| **xfce-extra/xfce4-verve-plugin** | Небольшая командная строка, встроенная в панель; быстрее, чем открывать другой терминал для выполнения команды. |
| **xfce-extra/xfce4-mount-plugin** | Монтирование устройств перечисленных в файле  по одному /etc/fstab клику. |
| **xfce-extra/xfce4-sensors-plugin** | Мониторинг аппаратных датчиков, таких, как температура центрального процессора, обороты вентилятора, температура жесткого диска, напряжение на материнской плате и другое. |
| **x11-themes/xfwm4-themes** | Several window-manager themes. |
| **xfce-extra/xfce4-power-manager** | Приложение для отображения и управления энергопотреблением (особенно важно для ноутбуков). Выберите режимы максимальной производительности или экономии заряда батареи. Отрегулируйте яркость экрана и настройте спящий режим, приостановку и действии при нажатии кнопки питания (то есть, когда крышка закрыта или нажата кнопка). Может предупреждать, когда батарея достигает определенного уровня или даже выключать машину. Включает в себя плагины панели для отображения состояния батареи/заряда и регулировки яркости. |

**6. Изучение и интеграция модуля в выбранную ОС**

Так как была выбрана ОС – Gentoo, то изучение и интеграция модуля была произведена еще при установке. К слову о основных понятиях и конфигурациях ядра.

**Основные идеи**

Общий процесс достаточно прост: предлагаются несколько возможностей на выбор, распределенные по категориям в отдельные меню и подменю, в которых можно выбрать поддержку аппаратного обеспечения и функций ядра, подходящие для системы.

Ядро включает *конфигурацию по умолчанию*, которая активирует отдельные части исходного кода ядра во время первого запуска команды menuconfig. В целом, предлагается широкий выбор разумных значений по умолчанию, что означает, что большинству пользователей потребуется только внести небольшое количество изменений в основную конфигурацию. Если необходимо отключить параметр, который был включен по умолчанию, убедитесь, что вы хорошо понимаете, что в точности делает данный параметр и последствия его отключения.

При конфигурации ядра Linux в первый раз, следует придерживаться умеренных настроек и вносить так мало изменений в основные настройки, насколько это возможно. В то же время, следует учитывать, что существуют определенные настройки, которые необходимо изменить, чтобы система действительно могла загружаться.

**Сравнение встроенных в ядро параметров и загружаемых модулей ядра**

Большинство параметров конфигурации могут находиться в трех состояниях: они могут быть как отключены вовсе (N), встроены прямо в ядро (Y), или собраны в качестве модулей (M). Модули хранятся вне ядра на файловой системе, в то время как встроенные компоненты встраиваются прямо в образ ядра.

Между [встроенными и модулями](https://wiki.gentoo.org/wiki/Kernel_Modules) существует важное различие: за некоторыми исключениями, ядро не предпринимает каких-либо попыток загрузить внешние модули, если они нужны системе; это задача пользователя решать, когда загружать модуль. В то время как определенные части системы могут иметь возможность загрузки модулей по требованию и доступны несколько утилит для автоматической загрузки модулей, рекомендуется включение поддержки аппаратного обеспечения и функций ядра непосредственно в само ядро. В этом случае, ядро может гарантировать функциональность и поддержку аппаратного обеспечения при необходимости. Это можно сделать, установив (Y) для каждой необходимой функции ядра. Для этого также требуется включить поддержку встраиваемого (firmware) программного обеспечения в ядре. Включите FW\_LOADER=y и CONFIG\_FIRMWARE\_IN\_KERNEL=y в конфигурационном файле ядра .config или следующее в **menuconfig**:

**Инструменты конфигурации**

Ядро предлагает несколько утилит для собственной конфигурации.

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| **make config** | Конфигуратор с текстовым интерфейсом. Вопросы следуют один за другим. На все вопросы должен быть дан ответ по порядку. Доступ к предыдущим вопросам невозможен. |
| **make menuconfig** | Конфигуратор основанный на ncurses с псевдографическим интерфейсом (возможен только ввод текста). Для изменения параметров есть навигация с помощью меню. |
| **make defconfig** | Создает новый конфигурационный файл с настройками по умолчанию, которые берутся из архитектурно-зависимых defconfig файлов. Используйте эту опцию, чтобы снова создать конфигурационный файл с настройками по умолчанию, так же как в архиве с исходным кодом. |
| **make nconfig** | Еще один конфигуратор с псевдографическим меню основанный на ncurses. Требуется, чтобы библиотека sys-libs/ncurses была установлена. |
| **make xconfig** | Конфигуратор с графическим интерфейсом основанный на Qt4. Должен быть установлен dev-qt/qtgui пакет. |
| **make gconfig** | Конфигуратор с графическим интерфейсом основанный на GTK+. Должны быть установлены пакеты x11-libs/gtk+, dev-libs/glib и gnome-base/libglade. |
| **make oldconfig** | Обзор изменений между версиями ядра, а так же обновление их для создания нового .config для ядра. |
| **make allyesconfig** | Включает все возможные опции ядра. Эта утилита устанавливает *все* опции ядра в \*. Убедитесь, что сохранена резервная копия конфигурации ядра, прежде чем будете экспериментировать с этой опцией!!! |

Существует несколько скриптов для создания смешанных конфигураций по умолчанию. Они могут помочь сделать более тонкую и в то же время более эффективную настройку. Выполните следующую команду для получения полного списка make targets:

**root #**make help

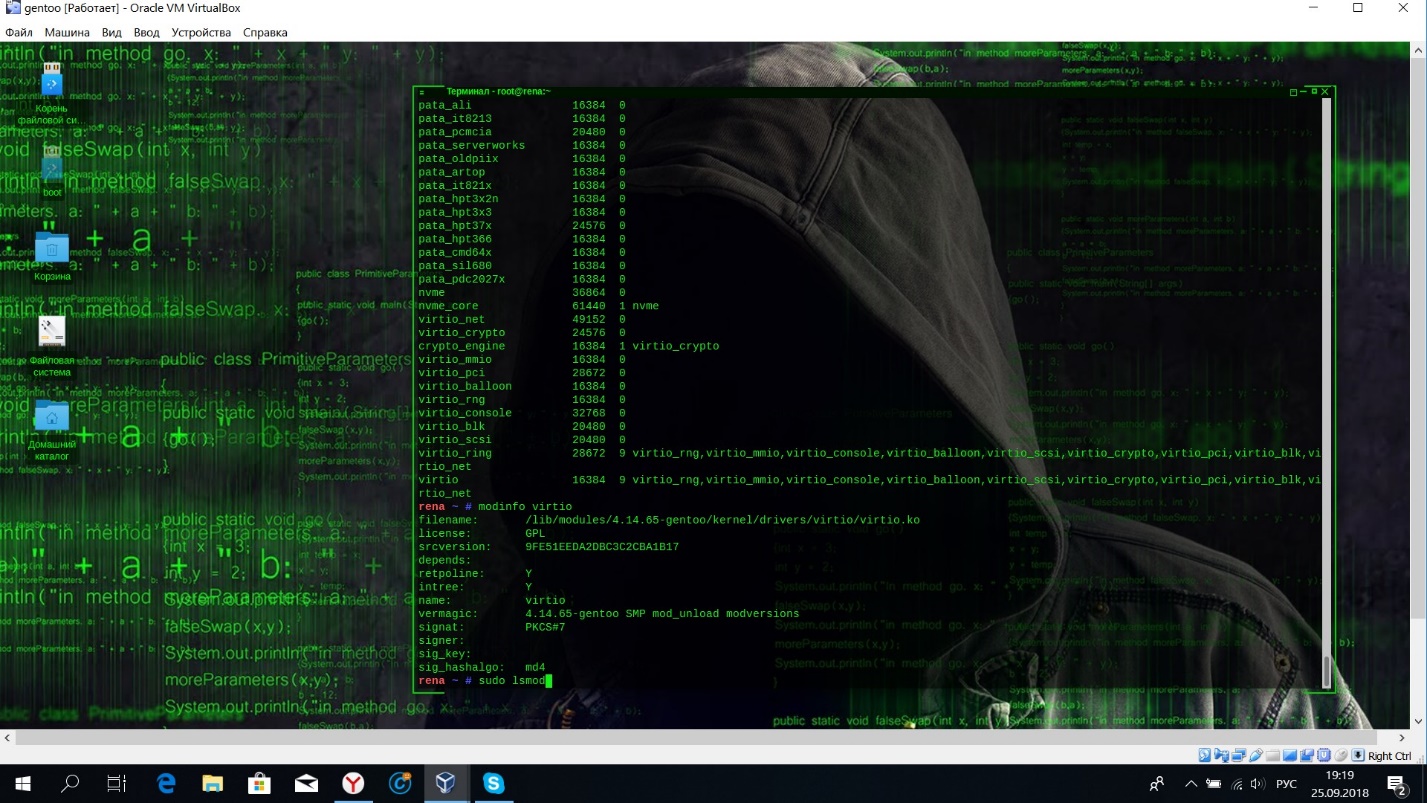
Если же вы хотите узнать какие у вас установлены модули, то воспользуйтесь командой:

**root #**lsmod

К примеру, можно получить информацию по какому-то установленному модулю используя команду:

**root #**Modinfo virtio

**Virtio** — **это** стандарт в операционной системе **Linux**для драйверов сетевых и дисковых устройств. При использовании **virtio** драйверы гостевой операционной системы “знают” что запущены в виртуальной среде и взаимодействуют с гипервизором.



Здесь вы можете увидеть файл модуля, его лицензию, автора и зависимости. Зависимости — это те модули, которые должны быть загружены для его нормальной работы. К сожалению, не для всех модулей доступно нормальное описание, но вы можете попробовать посмотреть описание зависимостей модуля.

**Интерпретация софта (пакетов, модулей)**

1. **Запись всего, что происходит в терминале в файл**

Фактически вы составляете полный лог всех команд, которые вы вводите в терминале и то, что выводится на экран.

Чтобы начать запись нужно выполнить команду script. Если вызвать команду script без параметров, то запись будет вестись в файл с именем typescript.

**root #**script

Скрипт запущен, файл - typescript

Файл будет создан именно в той директории, в которой вы находитесь. Вывести название текущей директории можно командой pwd. Если файл уже существует, то он будет очищен.

В качестве параметра команде script можно указать имя файла, в который вы хотите записывать всю активность в терминале. Пример вызова:

**root #**script myterminal.log

После ввода команды будет создан файл, но данные в него будут записаны только после того, как вы введете команду exit или нажмете сочетание клавиш Ctrl+D.

Зачем оно надо, если есть .bash\_history? Все просто, .bash\_history хранит только список введенных команд, а эта штука логирует всю активность в консоли.

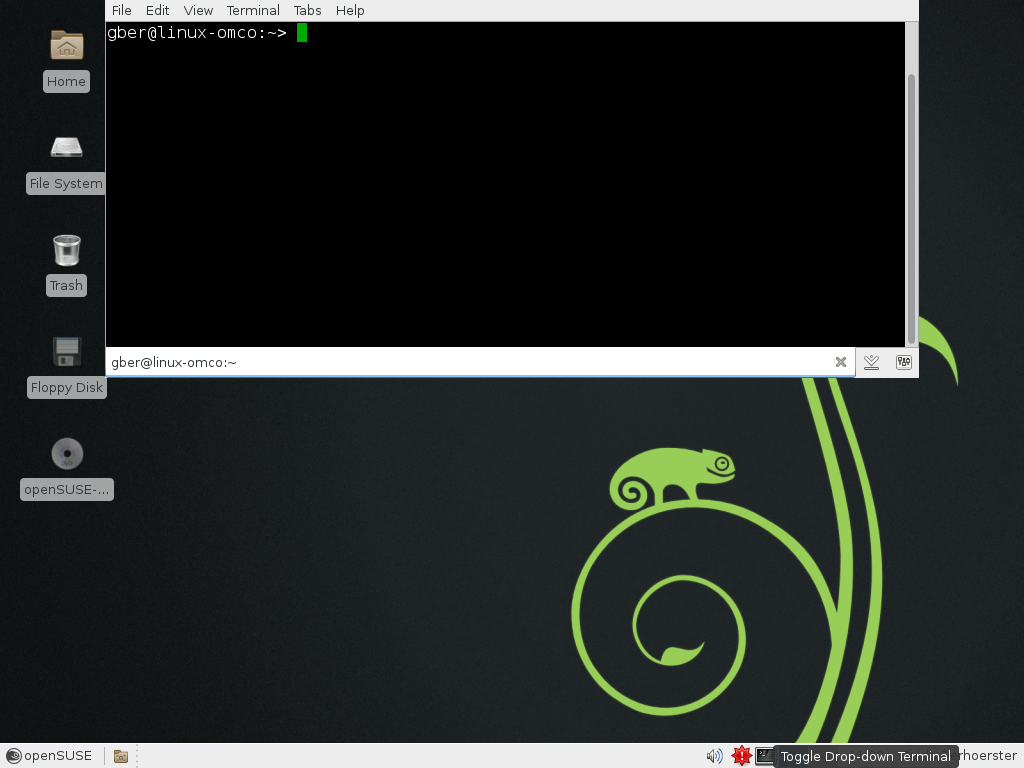
Было добавлено, за регистрирование сеанса gnome-terminal в файл:

bash -l -c 'dt=`date "+%Y%m%dx%H%M%S"`; read -p "Hostname> " host; echo -ne "\033]0;$host\007"; script -a -f -q -c "bash -l -c \"ssh $host\"" $HOME/sessionlogs/$host.sessionlog.$dt.txt'

Итак, некоторые разъяснения. Если вы хотите автоматическое протоколирование сессии для gnome-terminal, то мое решение абсолютно применимо, так как я использую его некоторое время, и он хранит каждый сеанс войти в отдельный файл. Это оказалось чрезвычайно полезным, но некоторые утверждают, что это не быстрее и проще чем вырезать и вставить. Решение состоит в том, чтобы просто регистрировать все время.

1. **Terminal**

Для установки выбран xfce4-terminal. Исходники были скачены с сайта: http://git.xfce.org/apps/xfce4-terminal/. После установки получаем:



Внешний вид довольно банальный, поэтому было принято решение о его изменении. В итоге у нас вышло:



1. **Работа с модулями ядра в Linux**

Для начального знакомства с техникой написания модулей ядра Linux создадим простейший модуль (код такого модуля интуитивно понятен всякому программисту), соберем и исполненим его. Ниже приведен код модуля:

main.c :

#include <linux/init.h>

#include <linux/module.h>

MODULE\_LICENSE( "GPL" );

MODULE\_AUTHOR( "Rena Ivis");

static int \_\_init hello\_init( void ) {

printk( "Hello, world!" );

return 0;

}

static void \_\_exit hello\_exit( void ) {

printk( "Goodbye, world!" );

}

module\_init( hello\_init );

module\_exit( hello\_exit );

Сборка модуля

Для сборки созданного модуля используем скрипт сборки Makefile:

Makefile :

CURRENT = $(shell uname -r)

KDIR = /lib/modules/$(CURRENT)/build

PWD = $(shell pwd)

DEST = /lib/modules/$(CURRENT)/misc

TARGET = main

obj-m := $(TARGET).o

default:

main -C $(KDIR) M=$(PWD) modules

clean:

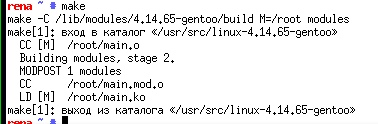
@rm -f \*.o .\*.cmd .\*.flags \*.mod.c \*.order

@rm -f .\*.\*.cmd \*.symvers \*~ \*.\*~ TODO.\*

@rm -fR .tmp\*

@rm -rf .tmp\_versions

Делаем сборку модуля:



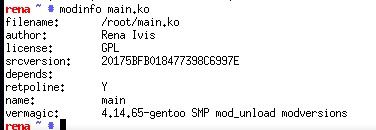
На этом модуль создан. Начиная с ядер 2.6 расширение файлов модулей сменено с \*.o на \*.ko:

$ ls \*.ko

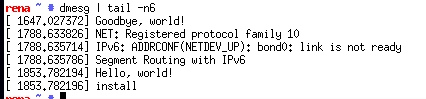
main.ko

Загрузка и исполнение

Модуль при загрузке/выгрузке выводит сообщение посредством вызова printk(). Этот вывод направляется на текстовую консоль. При работе в терминале X11 вывод не попадает в терминал, и его можно видеть только в лог файле /var/log/messages. Но и в текстовую консоль вывод направляется не непосредственно, а через демон системного журнала, и выводится на экран только если демон конфигурирован для вывода таких сообщений.



С помощью команды dmesg посмотрим работу модуля.



Можно увидеть, что после начала установки модуля, сработала команда printk("Hello, world!"), а при удалении млдуля было выведенно сообщение Goodbye, world!

**Поиск необходимого модуля**

Модули хранятся в каталоге "/lib/modules/<версия ядра>" в виде файлов с расширением «ko». Для получения списка всех модулей из дерева можно выполнить команду поиска всех файлов с расширением «ko» в каталоге с модулями текущего ядра:

find /lib/modules/`uname -r` -name ‘\*.ko’

Более детальную информацию о модуле можно получить при помощи команды modinfo:

# modinfo <модуль>

**Загрузка и выгрузка модулей**

Загрузить модуль в ядро можно при помощи двух команд: «insmod» и «modprobe», отличающихся друг от друга возможностью просчета и удовлетворения зависимостей. Команда «insmod» загружает конкретный файл с расширением «ko», при этом, если модуль зависит от других модулей, еще не загруженных в ядро, команда выдаст ошибку, и не загрузит модуль. Команда «modprobe» работает только с деревом модулей, и возможна загрузка только оттуда по имени модуля, а не по имени файла. Отсюда следует область применения этих команд: при помощи «insmod» подгружается файл модуля из произвольного места файловой системы (например, пользователь скомпилировал модули и перед переносом в дерево ядра решил проверить его работоспособность), а «modprobe» — для подгрузки уже готовых модулей, включенных в дерево модулей текущей версии ядра.

1. **Cowsay**

Приложение на Perl, которое при запуске выводит изображение говорящей или думающей коровы, нарисованной в ASCII символах, с «облачком» фразы или мыслей в котором написан какой‑нибудь заданный программе текст.

Пример: user@linux:~> uptime | cowsay -f tux

user@linux:~> uptime | cowsay -f tux

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/ 00:09 up 2:18, 4 \

| пользователя, |

| средняя |

| загруженность: 3,57, 3,30, |

\ 2,80 /

-----------------------------------------

\

\

.--.

|o\_o |

|:\_/ |

// \ \

(| | )

/'\\_ \_/`\

\\_\_\_)=(\_\_\_/

Был установлен базовый пакет, который в последствии претерпео некоторые изменения в исходно коде. Добавлены новые модули рисунков для вывода в консоль и изменена опция вызова приложения.

|  |
| --- |
| %opts = ( |
|  | 'e' => 'oo', |
|  | 'f' => 'default.cow', |
|  | 'n' => 0, |
|  | 'T' => ' ', |
|  | 'W' => 40, |
|  | ); |

Само приложение было установлена через написанный script который позаимствован с крупнейшего веб-сервиса для хостинга IT-проектов Github (https://github.com/schacon/cowsay/blob/master/install.sh).

**Заключение**

В Gentoo linux мы открыли для себя возможности тонкой настройки системы под конечного пользователя. Исходя из выше написанного можно сделать вывод что Gentoo действительно гибкая система. За счет флагов мы можем уменьшить бинарный файл, как следствие он будет быстрее выгружаться в оперативную память, и меньше занимать места на жестком диске. Тонкая настройка /etc/portage/make.conf дает заметный прирост в производительности.

Система Portage позволяет использовать несколько версий пакетов одновременно, что позволяет работать с самым стабильным ПО, даже если оно использует тестовые библиотеки. Так же мы рассмотрели несколько нюансов сборки ядра, и узнали про пару удобных инструментов для администрирования Gentoo.

Благодаря сложной установке и высокому порогу вхождения, gentoo обладает очень хорошей документацией и вики, что гарантирует, что пользователь, как минимум, натолкнется на существование большого количество дополнительного функционала до того, как это понадобятся. А это избавляет от стресса, который возникает при необходимости решить задачу, методов решения которой ты даже не представляешь. Рано или поздно в любом дистрибутиве ты попадешь в консоль. Приятно иметь под рукой привычные инструменты, способные решить твою проблему, а не искать их на форумах.

**Используемые ресурсы**

1. <https://habr.com/post/335022/> - Разрушители легенд — Gentoo Linux (Дата обращения: 17.09.2018)

2. <https://habr.com/post/201410/> - В чем прелесть Gentoo: Мифы и реальность (Дата обращения: 17.09.2018)

3. <https://losst.ru/nastrojka-gentoo> - ОБЗОР GENTOO LINUX (Дата обращения: 17.09.2018)

4. <https://wiki.gentoo.org/wiki/Handbook:AMD64> - Gentoo AMD64 Handbook (Дата обращения: 17.09.2018)

5. <https://ru.wikipedia.org/wiki/Cinnamon> - Cinnamon (Дата обращения: 17.09.2018)

6. <https://vikidalka.ru/2-116363.html> - Ознакомление с семейства ОС Linux (Дата обращения: 17.09.2018)

7. <https://www.linux.org.ru/tag/gentoo?section=3&offset=40> - Gentoo (Галерея) (Дата обращения: 17.09.2018)

8. <http://citforum.ru/operating_systems/linux/lig-1.shtml> - Введение в LINUX (Дата обращения: 17.09.2018)

9. <https://ru.wikipedia.org/wiki/Xfce> - Xfce (Дата обращения: 27.09.2018)

10. [https://github.com/schacon/cowsay/blob/master/install.sh](https://github.com/schacon/cowsay/blob/master/install.sh%20)  - cowsay (Дата обращения: 27.09.2018)

11. [http://uc.org.ru/sites/default/files/add-files/LDD\_programming\_kernel\_modules\_Linux.pdf](http://uc.org.ru/sites/default/files/add-files/LDD_programming_kernel_modules_Linux.pdf%20)  - Руководство по программированию модулей ядра Linux (Дата обращения: 27.09.2018)

12. [https://wiki.gentoo.org/wiki/NVidia/nvidia-drivers/ru#.D0.A3.D1.81.D1.82.D0.B0.D0.BD.D0.BE.D0.B2.D0.BA.D0.B0](https://wiki.gentoo.org/wiki/NVidia/nvidia-drivers/ru#.D0.A3.D1.81.D1.82.D0.B0.D0.BD.D0.BE.D0.B2.D0.BA.D0.B0 )  - NVidia/nvidia-drivers (Дата обращения: 27.09.2018)