Оглавление

Часть I. Основы UNIX

История возникновения и стандарты

История создания

В 1965 году Bell Telephone Laboratories (подразделение AT&T) совместно с General Electric Company и Массачусетским технологическим институтом (MIT) начали разрабатывать новую операционную систему, названную MULTICS (MULTiplexed Information and Computing Service). Перед участниками проекта стояла цель создания многозадачной операционной системы разделения времени, способной обеспечить работу нескольких сотен пользователей. От Bell Labs в проекте приняли участие два участника – Кен Томпсон (Ken Tompson) и Дэннис Ритчи (Dennis Ritchie) . Хотя система MULTICS так и не была завершена (в 1969 году Bell Labs вышла из проекта), она стала предтечей операционной системы, впоследствии получившей название Unix.

Однако Томпсон, Ритчи ряд других сотрудников продолжили работу над создание удобной системы программирования. Использую идеи и разработки, появившиеся в результате работы над MULTICS, они создали в 1969 году небольшую операционную систему, включавшую в себя файловую систему, подсистему управления процессами и небольшой набор утилит. Система была написана на ассемблере и применялась на компьютере PDP-7. Эта операционная система получила название UNIX, созвучное MULTICS и придуманное другим членом группы разработчиков, Брайаном Керниганом (Brian Kernigan).

Хотя ранняя версия UNIX много обещала, она не смогла бы реализовать весь свой потенциал без применения в каком-либо реальном проекте. И такой проект нашелся. Когда в 1971 году патентному отделу Bell Labs понадобилась система обработки текста, в качестве операционной системы была выбрана UNIX. К тому времени она была перенесена на более мощный PDP-11, да и сама немного подросла: 16K занимала собственно система, 8K отводилось прикладным программам, максимальный размер файла был установлен в 64K при 512K дискового пространства.

Вскоре после создания первых ассемблерных версий Томсон начал работать над компилятором для языка FORTRAN, а в результате разработал язык B. Это был интерпретатор со всеми свойственными интерпретатору ограничениями, и Ритчи переработал его в другой язык, названный C, позволявший генерировать машинный код. В 1973 году ядро операционной системы было переписано на языке высокого уровня C, – неслыханный до этого шаг, оказавший громадное влияние на популярность UNIX. Это означало, что теперь система UNIX может быть перенесена на другие аппаратные платформы за считанные месяцы и внесение изменений не представляло особых трудностей. Число работающих UNIX-систем в Bell Labs превысило 25, и для сопровождения UNIX была сформирована группа UNIX System Group (USG).

Исследовательские версии(AT&T Bell Labs)

В соответствии с федеральным законодательством США, AT&T не имела права коммерческого распространения UNIX и использовала ее для собственных нужд, но, начиная с 1974 года, операционная система стала передаваться университетам для образовательных целей.

Операционная система модернизировалась, каждая новая версия снабжалась соответствующей редакцией Руководства Программиста, откуда и сами версии получили название редакций (Edition). Всего с 1971 по 1989 год было выпущено 10 редакций. Ниже перечислены наиболее важные редакции.

Редакция 1 (1971)

Первая версия UNIX, написанная на ассемблере для PDP-11. Включала в себя язык B и много известных комманд и утилит, в том числе **`cat, chdir, chmod, cp, ed, find, mail, mkdir, mkfs, mount, mv, rm, rmdir, wc, who`**.
В основном использовалась как инструментальное средство обработки текстов для патентного отдела Bell Labs.

Редакция 3 (1973)

В системе появилась команда **`cc`**, запускавшая компилятор языка C. Число установленных систем достигло 16.

Редакция 4 (1973)

Первая система, в которой ядро написано на языке высокого уровня C.

Редакция 6 (1975)

Первая версия UNIX, доступная за пределами Bell Labs. Система полностью переписана на языке C. С этого времени начинается появление новых версий, разработанных не в Bell Labs и рост популярности UNIX. Эта версия системы была установлена в Калифорнийском университете в Беркли, и на ее основе вскоре была выпущена первая версия BSD (Berkeley Software Distributuion) UNIX.

Редакция 7 (1979)

Включала в себя командный интерпретатор Bourne Shell и компилятор C от Кернигана и Ритчи. Ядро системы было переписано для переносимости на другие платформы. Лицензия на эту версию была куплена фирмой Microsoft, которая разработала на ее базе операционную систему XENIX.

Популярность UNIX росла, и к 1977 году число работающих систем превысило 500. В этом же году система впервые была портирована на компьютер, отличный от PDP.

Генеалогия UNIX

Не существует некоторой «стандартной» системы UNIX, все UNIX-подобные системы имеют характерные только для них особенности и возможности. Но за разными названиями и особенностями все же нетрудно заметить архитектуру, пользовательский интерфейс и среду программирования UNIX. Объясняется это достаточно просто – все эти операционные системы являются ближними или дальними родственниками. Ниже описаны наиболее яркие представители данного семейства.

System III (1982)

Не желая терять инициативу по развитию UNIX, AT&T в 1982 году объединила несколько существующих версий ОС и создала версию под названием System III. Данная версия была предназначена для распространения за пределами Bell Labs и AT&T, и положила начало мощной ветви UNIX, которая и сегодня жива и развивается.

System V (1983)

В 1983 году выпущена System V, а позже – еще несколько релизов (Release) к ней:

- SVR2 (1984): InterProcess Communication (IPC) разделяемая память, семафоры
- SVR3 (1987): Система I/O Streams, File System Switch, разделяемые библиотеки
- SVR4 (1989): NFS, FFS, сокеты BSD. SVR4 объединила возможности нескольких известных версий UNIX – SunOS, BSD UNIX и предыдущих релизов System V. Многие компоненты этой системы были поддержаны стандартами ANSI, POSIX, X/Open и SVID.

UNIX BSD (1978) (На основе 6-й редакции UNIX)

- 1981 по заказу DARPA в BSD UNIX был встроен стек TCP/IP (в 4.2BSD)
- 1983 активно использовала сетевые технологии и могла подключаться к сети ARPANET
- 1986 выпущена версия 4.3BSD
- 1993 выпущены 4.4BSD и BSD Lite (последние выпущенные версии).

OSF/1 (1988) (Open Software Foundation)

- В 1988 году IBM, DEC, HP объединились с целью создания независимой от AT&T и SUN версии UNIX и создали организацию под названием OSF. Результатом деятельности этой организации стала операционная система OSF/1.

Стандарты

Чем больше появлялось различных вариантов UNIX, тем очевиднее становилась необходимость стандартизации системы. Наличие стандартов облегцает переносимость приложений и защищает как пользователей, так и производителей. В результате возникло несколько организаций, связанных со стандартизацией, и был разработан ряд стандартов, оказывающих влияние на развитие UNIX.

IEEE POSIX (Institute of Electrical and Electronics Engineers Portable Operating System Interface)

- 1003.1 (1988) стандартизация API (Application Programming Interface) ОС
- 1003.2 (1992) определение командного интерпретатора и утилит
- 1003.1b (1993) API приложений реального времени
- 1003.1c (1995) определений “нитей” (threads)

ANSI (American National Standards Institute)

- Стандарт X3.159 (1989)

- Синтаксис и семантика языка C
- Содержимое стандартной библиотеки libc

X/Open

- 1992 стандарт Xwindow
- 1996 создание совместно с OSF прользовательского интерфейса CDE (Common Desktop Environment) и его сопряжение с графической оболочкой Motiff

SVID (System V Interface Definition)

Описывает внешние интерфейсы UNIX версий System V. В дополнение к SVID был выпущен SVVS (System V Verification Suite) – набор текстовых программ, позволяющий определить, соответствует ли система стандарту SVID и достойна ли она носить гордое имя System V.

Известные версии UNIX

- IBM AIX на базе SVR2 со многими чертами SVR4, BSD, OSF/1
- HP-UX версия фирмы HP
- IRIX версия фирмы Silicon Graphics, похожа на SVR4
- Digital UNIX версия фирмы DEC на основе OSF/1
- SCO UNIX (1988) одна из первых UNIX систем для PC разработанная на основе SVR3.2
- Solaris версия UNIX SVR4 компании Sun Microsystems

История FreeBSD

Разработка FreeBSD началась в 1993 году с быстрорастущего набора патчей пользователей системы 386BSD. Этот набор позже вырос и отделился от 386BSD в отдельную операционную систему, включив в себя код от Free Software Foundation. Первая официальная версия FreeBSD 1.0 вышла в декабре 1993 года. Walnut Creek CDROM согласилась распространять FreeBSD на компакт-диске и также предоставила для работы проекту отдельный компьютер с интернет-соединением. Затем, в мае 1994 года, последовал успешный выпуск FreeBSD 1.1.

Однако, из соображений законности использования исходных кодов BSD Net/2 в 386BSD, команда разработчиков FreeBSD переработала большую часть системы ко времени выпуска FreeBSD 2.0 в январе 1995 года, используя 4.4BSD-Lite. Руководство к FreeBSD содержит более подробную историческую информацию о происхождении системы.

Версии системы FreeBSD

На 4 мая 2009 года последний релиз FreeBSD имеет номер 7.2.

FreeBSD 4

4.0-RELEASE появилась в марте 2000 года и последняя версия 4.11 была выпущена в январе 2005 года. FreeBSD 4 была очень популярной у провайдеров интернет и хостингов во время первого «пузыря доткомов» и считалась одной из самых стабильных и высокопроизводительных систем класса Unix. До сих пор в интернете можно найти серверы с FreeBSD 4, которые обслуживают миллионы запросов изо дня в день.

Одним из главных недостатков FreeBSD 4 считается плохая поддержка нескольких процессоров, особенно в режиме многопоточности.

FreeBSD 4 поставила своеобразный рекорд по продолжительности разработки одной ветки операционной системы — за пять лет были устранено большое количество ошибок и получена на редкость стабильная система.

В середине разработки FreeBSD 4 от нее отпочковался проект DragonFlyBSD, основатели которого положили своей целью серьезную оптимизацию ядра для высоконагруженных систем, в частности лучшую поддержку многопроцессорности (уменьшение времени, необходимого для переключения потоков и пр.).

FreeBSD 5

Через 3 года разработки, в январе 2003 года, была выпущена долгожданная версия 5.0-RELEASE. Эта версия предоставляла расширенную поддержку многопроцессорности и многопоточности, а также поддержку платформ UltraSPARC и IA-64.

Наибольшие архитектурные изменения в FreeBSD 5 — это изменение механизма блокировки на нижнем уровне ядра, чтобы улучшить поддержку многопроцессорных SMP-систем. Это освободило большую часть ядра от так называемой «гигантской блокировки» (Giant lock). Теперь в ядре появилась возможность выполнять более одной задачи одновременно. Другим важным изменением была реализация «родной» поддержки многопоточности типа M:N под названием Kernel Scheduled Entities (KSE). Начиная с FreeBSD 5.3 эта реализация потоков была установлена по-умолчанию, пока не была заменена

на реализацию модели 1:1 во FreeBSD 7.

В FreeBSD 5 была серьёзно изменена система блочного ввода-вывода посредством введения модульной структурной системы преобразования запросов ввода-вывода GEOM (внесённой Poul-Henning Kamp). GEOM даёт возможность создавать различную функциональность, такую как зеркалирование (mirroring) или шифрование.

Версии 5.4 и 5.5 были признаны стабильными и высокопроизводительными, но более ранние версии не годились для использования в рабочих условиях.

FreeBSD 6

FreeBSD 6.0 была выпущена 4 ноября 2005 года. 11 ноября 2008 года была выпущена версия 6.4. Эти версии являются продолжением оптимизации поддержки SMP и многопоточности вкупе с расширенной поддержкой стандарта 802.11, записью событий безопасности проекта TrustedBSD, серьезными улучшениями производительности сетевой подсистемы. Основное достижение этого релиза — исключение «гигантской блокировки» (Giant lock) из виртуальной файловой подсистемы (VFS), реализация дополнительной, более производительной поддержки многопоточности (libthr) с моделью 1:1, и добавление OpenBSM — первичного модуля безопасности, который был создан проектом TrustedBSD.

FreeBSD 7

FreeBSD 7.0 выпущена 27 февраля 2008 года. 5 января 2009 года вышла версия 7.1. Новое в этой ветке включает в себя: оптимизированный сетевой протокол транспортного уровня SCTP, журналирование в файловой системе UFS2, экспериментальная адаптированная версия файловой системы ZFS (разработанной компанией Sun), компилятор GCC4.2, базовая поддержка платформы ARM, новый менеджер памяти jemalloc, оптимизированный для параллельных вычислений[6], и большие изменения и оптимизации подсистем работы с сетями, аудио-устройствами и SMP-системами.[7] Новая система показала значительные улучшения в скорости по сравнению с предыдущими версиями и системой Linux.

4 мая 2009 года вышла версия 7.2. Нововведения в этой версии: поддержка семейства процессоров UltraSPARC III ("Cheetah") и SPARC64; возможность назначения нескольких IPv4- и IPv6-адресов каждой клетке — виртуальной машине Jail; реализация техники Superpages, прозрачного для приложений увеличения размера (с 4КБ до 4МБ) страниц виртуальной памяти; увеличенное до 6 Гб адресное пространство ядра для 64-разрядных процессоров; включена поддержка множественных таблиц маршрутизации, в том числе для клеток; улучшена совместимость в работе 32-разрядных клеток в 64-разрядном окружении; из NetBSD портирован демон btpand с реализацией поддержки профилей Bluetooth Network Access Point (NAP), Group Ad-hoc Network (GN) и Personal Area Network User (PANU); добавлен новый драйвер sdhci с поддержкой PCI-SD хост-контроллеров (кард-ридеров); обновлен модуль ядра DRM (Direct Rendering Manager) в котором улучшена поддержка графических процессоров (GPU) AMD/ATI, XGI, Intel; обновлены драйверы сетевых и дисковых устройств. В скором времени ожидается разработка видеодрайвера NVIDIA для 64-разрядной архитектуры amd64. Окончательная адаптация файловой системы ZFS v.13 для этой ветки почти завершена.

FreeBSD 8

7 июля 2009 года выпущена первая публичная бета версия FreeBSD 8.0, первый кандидат в релизы доступен для пользователей 21 сентября 2009 года, второй кандидат в релизы

планируется на 14 октября, третий кандидат в релизы на 28 октября. Релиз будет объявлен 5 ноября 2009 года.

Версия 8.0 включает в себя большое количество новой функциональности, такой как:

- Система Dtrace (фреймворк динамической трассировки, для выявления неправильной работы ядра и приложений на работающей системе в режиме реального времени), взятая от Sun из Solaris 10 (включена и работает в версии 7.2).
- Поддержка Xen DomU.
- Виртуализация сетевой поддержки.
- Улучшенная поддержка ZFS.
- Новая подсистема USB.

Модель разработки FreeBSD

Существует около 4000 разработчиков, которые работают над системой на добровольной основе. Все они могут читать дерево репозитория, но не могут вносить изменения. Вместо этого разработчик обращается к коммиттеру , который имеет право вносить изменение в код. Существует около 400 коммитеров. Разработчик может вырасти по социальной лестнице проекта и стать коммитером. Кандидатуру нового коммитера предлагает к рассмотрению ментор будущего коммитера. В зависимости от основной области деятельности, новый коммитер утверждается основной командой, portmgr@ или docmgr@. Основная команда является административным ядром проекта и состоит из 9 человек, которые выбираются на 2 года коммитерами из коммитеров. Основная команда решает конфликты между коммитерами.

Участники проекта разрабатывают ветку CURRENT («текущая» версия) и несколько STABLE («стабильная», стабильность означает гарантию неизменности интерфейсов, как то API, ABI и так далее).

Новый код помещают в ветку CURRENT, где он получает более широкое тестирование. Новые функции, добавленные в CURRENT, могут остаться в системе или от них могут отказаться, если реализация окажется неудачной. Иногда эта версия может оказаться в непригодном для использования состоянии. С началом использования perforce как вспомогательного репозитория, и с выделением projects/ области в svn, проект стремится гарантировать постоянную работоспособность CURRENT.

STABLE-версия содержит только те нововведения, которые прошли проверку в CURRENT. Тем не менее, эта версия тоже предназначена, в основном, для разработчиков. Не рекомендуется обновлять ответственные рабочие серверы до STABLE, предварительно её не протестировав. На основе STABLE регулярно создаются тщательно протестированные разработчиками, группой release-инженеров и более широким кругом пользователей RELEASE-версии.

После выпуска релизов создаются дополнительные ветви разработки для поддержки релизов, но в них вносятся лишь самые необходимые изменения, исправляющие серьёзные ошибки или проблемы с безопасностью системы. До четвёртой версии FreeBSD у стабильной и текущей веток был один и тот же старший номер версии. Затем текущей ветви был присвоен номер 5, а у стабильной остался номер 4.

В настоящее время стабильная версия имеет номер 7, а текущая — 8. Группа разработчиков, исправляющих проблемы безопасности системы (security officers) поддерживает ветвь 6-STABLE для тех пользователей, которые ещё не обновили FreeBSD до версии 7.

Концепции UNIX

Физическая организация системы

- Ядро взаимодействует с аппаратной частью и предоставляет посредством системных вызовов услуги ввода/вывода, создания и управления процессами.
- Приложения (системные и прикладные) обеспечивает пользовательский интерфейс.

Логическая организация системы

Деятельность системы сводится к обработке *файлов процессами*. То есть, с точки зрения прикладного процесса, все в системе является *файлом*. Через унифицированный интерфейс файловой системы осуществляется доступ к различным файлам в файловой системе и *устройствам*: терминалам, принтерам, стримерам, сети и даже к памяти.

Идеология использования системы

Наличие простых, но мощных хорошо документированных интерфейсов, в том числе интерфейсов пользователя. Имея в своем распоряжении набор утилит, каждая из которых решает узкую специализированную задачу, вы можете конструировать из них сложные комплексы для решения практически сколь угодно сложных задач.

Пример используемых программ для построения почтового сервера:

`sendmail` — транспортный агент (протокол SMTP)
`dovecot` — агент доступа (протоколы POP/IMAP)
`clamav` — антивирус
`SpamAssassin` — средство борьбы со спамом

Пример использования данной концепции в командной строке. Задача – найти 5 домашних каталогов пользователей, использующих больше всего дискового пространства:

```
# du -sk /home/* | sort -nr | head -n5
```

Подключение к UNIX

ОС UNIX изначально создавалась как многопользовательская операционная система, поэтому она обладает широкими коммуникационными возможностями и предлагает различные варианты терминального подключения пользователей:

Текстовые терминалы

- Интерфейс **`RS-232`**
- Интерфейс **`D-SUB PS/2`**
- Протокол **`telnet`**
- Протокол **`ssh`**

Графические терминалы

- Протокол **`Xwindow`**

Структура команд UNIX

Прежде чем выполнить команду, шелл производит определенную последовательность действий:

- Анализирует синтаксис команды. Если обнаружена синтаксическая ошибка, выводится соответствующее сообщение. При этом шелл анализирует командную строку в соответствии с синтаксисом собственного языка, а не семантику вызова конкретной команды, например, наличие тех или иных аргументов.
- Производит некоторые подстановки:
  - Заменяет указанные переменные их значениями. Например, если значение переменной **`var`** равно **`/usr/bin`** то при вызове команды
    `find $var -name firefox -ls`
    фактически будет запущена команда
    `find /usr/bin -name firefox -ls`
  - Формирует списки файлов, заменяя шаблоны. При этом производится подстановка следующих шаблонов:

    `*` – соответствует любому имени файла (или его части), кроме скрытых файлов (это файлы, имя которых начинается с точки)
    `[abc]` – соответствует *одному* любому символу из перечисленных в квадратных скобках (здесь: **`a`** *либо* **`b`** *либо* **`c`**)
    `?` – соответствует любому одиночному символу
- Делает соответствующие переназначения потоков ввода/вывода, если указаны соответствующие символы перенаправления (**`>`**, **`<`**, **`>>`**, **`|`**)
- Выполняет команду, передавая ей аргументы с выполненными подстановками. При этом:
  - Если команда является заранее определенной (например, пользователем) функцией, вызывается эта функция
  - Если команда является встроенной командой шелла, выполняется эта встроеннная команда
  - В противном случае производится поиск программы в каталогах, перечисленных в переменной окружения **`PATH`**, если имя программы указано без пути. Если команда содержит элементы пути (относительный или абсолютный путь), производится запуск программы по указанному пути. Если программа не найдена, выводится сообщение об ошибке.

Общий способ использования команд выглядит следующим образом:

`<команда> <ключи> <аргументы>`

- команда (программа) – определяет необходимое действие (*что делать?*)
- ключи (опции) – *как делать?*
- аргументы – *с чем делать?*

Формат ключей

Ключи без параметров:

-a
-a -l *или* **-al** *или* **-la**

Ключи с параметрами:
-i fxp0
-ni fxp0
-nifxp0

Длинные ключи:
--prefix=/usr/local
--exclude /usr/ports

Примеры использования команды ls

```
ls -a
ls -al
ls -l /bin
```

Исключения

Некоторые команды имеют формат указания ключей и аргументов, отличающийся от стандартного:

```
find /usr/share -name '*html'
ps ax
tar xf archive.tgz
```

Документация UNIX

Основным источником информации почти по любому вопросу в UNIX являются страницы экранной документации – **manual pages**, которые можно просмотреть с помощью команды **man**.

Использование команды man

Для отображения информации на экране программа **man** использует т.н. пейджер (**pager**) **more** или **less**. Соответственно, при просмотре информации (для навигации по справочной странице, поиске и т.д.) используются команды запущенного пейджера. Получить помощь по этим командам всегда можно, нажав клавишу **h**. Выход из пейджера осуществляется с помощью команды **q**. Поиск информации осуществляется с помощью команд **/** (вперед) и **?** (назад), после которых указывается шаблон для поиска.

Для получения более подробной справки по встроенной документации в UNIX используйте команду

```
man man
```

Разделы man страниц

Справочная информация делится на разделы. Вот список разделов, характерный для системы FreeBSD 7.2:

```
1 FreeBSD General Commands Manual
2 FreeBSD System Calls Manual
3 FreeBSD Library Functions Manual
4 FreeBSD Kernel Interfaces Manual
5 FreeBSD File Formats Manual
6 FreeBSD Games Manual
7 FreeBSD Miscellaneous Information Manual
8 FreeBSD System Manager's Manual
9 FreeBSD Kernel Developer's Manual
```

Иногда страницы с одинаковыми именами могут располагаться в нескольких разделах. В этом случае для получения нужной информации необходимо указывать номер нужного раздела. Пример:

```
man passwd
man 5 passwd
```

Секции man страниц

Страницы справки поделены на секции, каждая из которых имеет название и содержит соответствующую информацию. На разных справочных страницах состав секций может быть различным, при этом три секции являются обязательными (**NAME**, **SYNOPSIS** и **DESCRIPTION**). Ниже приведены наиболее часто встречающиеся секции:

NAME	– имя страницы с очень кратким описанием предмета (одна строка)
SYNOPSIS	– синопсис, краткий обзор
DESCRIPTION	– полное описание предмета справки
ENVIRONMENT	– используемые переменные окружения
EXIT STATUS	– коды возврата программ
EXAMPLES	– примеры использования
SEE ALSO	– рекомендуемые **man** страницы для получения дополнительной информации

Поиск man страниц

Для поиска нужной страницы справочного руководства используются команды **whatis** (аналог **man -f**) и **apropos** (аналог **man -k**). Обе команды производят поиск по индексу, составленнму из секций **NAME** всех справочных страниц. При этом команда **whatis** ищет целое слово, переданное в качестве параметра, а **apropos** – использует параметр как шаблон для поиска (расширенное регулярное выражение) и игнорирует границы слов. Для построения нового индекса (например, при изменении состава **man** страниц) используется команда **makewhatis**.

Структура файловой системы

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

UNIX не использует букв дисков, или других имен дисков в пути. Это значит, что не нужно писать **`c:/foo/bar/readme.txt`** в UNIX.

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

Точкой монтирования называется каталог, который будет соответствовать корню смонтированной файловой системы. Стандартные точки монтирования включают /usr, /var, /tmp, /mnt и /cdrom. Эти каталоги обычно перечислены в файле /etc/fstab, в котором указаны файловые системы и их точки монтирования. Большинство файловых систем, описанных в /etc/fstab монтируются автоматически, если только для них не указана опция noauto.

Монтирование файловых систем во FreeBSD

Для монтирования файловых используется команда **`mount`**, для размонтирования – **`umount`**. Подробную информацию об этих командах и используемых ими опциях можно найти справочной системе.

Файловые системы содержатся в разделах. Каждый раздел обозначается буквой от **`a`** до **`h`**. Каждый раздел может содержать только одну файловую систему, это значит что файловая система может быть описана ее точкой монтирования в файловой иерархии, или буквой раздела, в котором она содержится.

FreeBSD также использует дисковое пространство под раздел подкачки (**`swap space`**). Подкачка позволяет FreeBSD работать с виртуальной памятью. Ваш компьютер может работать так, как если бы у него было больше памяти, чем есть на самом деле. Когда у FreeBSD кончается память, она перемещает часть данных, не используемых в данный момент, в раздел подкачки и возвращает их обратно (перемещая в подкачку что-то другое), когда они нужны.

По некоторым разделам есть определенные соглашения:

Раздел	Соглашение
`a`	Как правило, содержит корневую файловую систему
`b`	Как правило, содержит раздел подкачки
`c`	Как правило, такого же размера, что и весь слайс (**`slice`**). Это позволяет утилитам, которым нужно работать над всем слайсом (например, сканер плохих блоков), работать с разделом **`c`**. В обычной ситуации не нужно создавать файловую систему на этом разделе.

Каждый раздел, содержащий файловую систему, хранится на том, что во FreeBSD

называется слайс (**slice**). Слайс -- это термин FreeBSD, то, что обычно называют разделом в других операционных системах (например, Windows и Linux). Слайсы нумеруются с 1 по 4.

Номера слайсов следуют за именем устройства, предваряемые строчной **s**, начиная с 1. Так "**da0s1**" это первый слайс первого SCSI устройства. Может быть только четыре физических слайса на диске, но могут быть логические слайсы нужного типа внутри физических слайсов. Эти дополнительные слайсы нумеруются начиная с 5, так что "**ad0s5**" это первый дополнительный слайс на первом IDE диске. Эти устройства используются файловыми системами, занимающими весь слайс.

Слайсы, "эксклюзивно выделенные (**dangerously dedicated**)" физические устройства и другие устройства содержат разделы, представляемые буквами от **a** до **h.** Эти буквы добавляются к имени устройства. "**da0a**" это раздел **a** на первом устройстве **da**, который "эксклюзивно выделен". "**ad1s3e**" это пятый раздел в третьем слайсе второго IDE диска.

Наконец, каждый диск имеет своё имя, которое начинается с кода, обозначающего тип диска, затем идет номер диска. В отличие от слайсов, нумерация дисков начинается с 0. Основные коды указаны в таблице ниже:

Код	Значение
ad	ATAPI (IDE) диск
da	SCSI direct access диск
acd	ATAPI (IDE) CDROM
cd	SCSI CDROM
fd	Floppy disk

Иерархия файловой системы

Полное описание иерархии файловой системы есть в справочном руководстве (**man hier**). В таблице ниже перечислены наиболее важные каталоги.

Каталог	Описание
/	Корневой каталог файловой системы.
/bin/	Основные утилиты, необходимые для работы как в однопользовательском, так и в многопользовательском режимах.
/boot/	Программы и конфигурационные файлы, необходимые для нормальной загрузки операционной системы.
/boot/defaults/	Конфигурационные файлы с настройками по умолчанию, используемые в процессе загрузки операционной системы
/dev/	Файлы устройств
/etc/	Основные конфигурационные файлы системы и скрипты.
/etc/defaults/	Основные конфигурационные файлы системы с настройками по умолчанию

/etc/mail/	Конфигурационные файлы для систем обработки почты
/etc/namedb/	Конфигурационные файлы для утилиты named
/etc/periodic/	Файлы сценариев, выполняемые ежедневно, еженедельно и ежемесячно
/etc/ppp/	Конфигурационные файлы для утилиты ppp
/mnt/	Пустой каталог, часто используемый системными администраторами как временная точка монтирования.
/proc/	Виртуальная файловая система, отображающая текущие процессы
/rescue/	Статически собранные программы для восстановления после сбоев.
/root/	Домашний каталог пользователя root.
/sbin/	Системные утилиты и утилиты администрирования, необходимые для работы как в однопользовательском, так и в многопользовательском режимах.
/tmp/	Временные файлы. Содержимое /tmp обычно теряется во время перезагрузки системы.
/usr/	Большинство пользовательских утилит и приложений.
/usr/bin/	Пользовательские утилиты и приложения общего назначения.
/usr/include/	Файлы стандартных библиотек.
/usr/libdata/	Файлы данных для различных утилит.
/usr/libexec/	Системные демоны и утилиты (выполняемые другими программами).
/usr/local/	Локальные пользовательские приложения, библиотеки, и т.д. Также используется по умолчанию коллекцией портов. Внутри /usr/local иерархия каталогов должна следовать стандарту иерархии для /usr. Исключение составляют каталог man, который расположен непосредственно в /usr/local, а не в /usr/local/share, и документация портов, которая расположена в share/doc/port.
/usr/obj/	Архитектурно-зависимые файлы и каталоги, образующиеся в процессе сборки системы из исходных текстов в /usr/src.
/usr/ports/	Коллекция портов FreeBSD (опционально).
/usr/sbin/	Системные утилиты и утилиты администрирования (исполняемые пользователем).
usr/share/	Архитектурно-независимые файлы.
/usr/src/	Исходные тексты BSD и/или программ.
/usr/X11R6/	Утилиты, приложения и библиотеки X11R6 (X Window System; необязательно).
/var/	Файлы журналов общего назначения, временные, перемещаемые файлы и файлы очередей.
/var/log/	Различные файлы системных журналов.
/var/mail/	Почтовые ящики пользователей.

/var/spool/	Файлы очередей печати, почты, и пр.
/var/tmp/	Временные файлы, которые обычно сохраняются во время перезагрузки системы, если только /var не является файловой системой в памяти.

Типы и виды файлов

В отличие от ОС Windows, в UNIX расширение файла не определяет его тип и может вообще отсутствовать. Расширение обычно используется для удобства пользователя и операционной системе не требуется. Вместо этого, UNIX определяет тип файла с помощью *магии* (подсистемы **magic**), которая на основании *магических чисел* (**magic numbers**), полученных в результате анализа содержимого файла делает вывод о типе файла. Таблица магических чисел постоянно обновляется и уточняется и содержит сигнатуры различных типов файлов. Кроме того, в UNIX существуют различные *виды* файлов, информация о которых содержится атрибутах файловой системы для этих файлов. Всего в UNIX существует 6 различных видов файлов:

- Обычный файл (**regular file**)
- Каталог (**directory**)
- Специальный файл устройства (**special device file**)
- FIFO, или именованный канал (**named pipe**)
- Символьная ссылка (**symbol link**)
- Сокет (**socket**)

Файлы устройств и каталог /dev

Термин "устройство" используется в основном по отношению к аппаратному обеспечению системы, такому как диски, принтеры, графические адаптеры, устройства ввода текста. В UNIX доступ к большинству этих устройств можно получить через специальные файлы устройств, расположенные в каталоге **/dev**. При добавлении в систему нового устройства или добавлении поддержки дополнительных устройств потребуется создать один или несколько файлов устройств для нового оборудования. Для устранения необходимости в этих действиях в FreeBSD используется **Device filesystem**, или **DEVFS**. Она предоставляет доступ к пространству устройств ядра через общую файловую систему. Вместо создания и модификации файлов устройств, **DEVFS** создает специальную файловую систему.

Основные команды для работы с файлами

Команды для работы с файловой системой

Просмотр файловой системы - ls

Команда **ls** (**list files**) предназначена для просмотра содержимого каталогов и получения информации о файлах. Примеры:

```
$ ls
$ ls -a
$ ls /bin
$ ls -l /
$ ls -l /dev
$ ls -l ..
$ ls -a .
```

Определение текущего каталога — pwd

pwd (**p**rint **w**orking **d**irectory) показывает текущий рабочий каталог.

Смена текущего каталога — cd

cd (**c**hange **d**irectory) меняет текущий рабочий каталог. Если команда **cd** вызвана без параметров, происходит переход в домашний каталог пользователя. Команда **cd -** позволяет вернуться в предыдущий каталог (каталог, откуда был осуществлен переход в текущий). Примеры:

```
$ cd /usr
$ cd bin
$ cd ..
$ cd
$ cd -
$ cd ~u1
```

Создание файла с помощью команды touch

Команда **touch** меняет временные метки файла и имеет побочное действие, которое используется гораздо чаще, чем основное – если файла с указанным именем нет, создается пустой файл.

Создание каталога — mkdir

Команда **mkdir** (**mk**e **dir**ectory) создает каталоги, в том числе и промежуточные (если указана опция -p). Примеры:

```
$ mkdir /tmp/test
$ mkdir -p /tmp/a/b/c
```

Удаление файла — rm

Команда **rm** удаляет файлы и каталоги, в т.ч. каталоги с содержимым (опция **-r** или **-R**). Опция **-f** подавляет вывод запросов (например, при удалении файлов, доступных только для чтения) и ошибок при удалении, а опция **-i** выводит запрос при удалении каждого файла. Примеры:

```
$ touch /tmp/test
$ rm -r /tmp/test
# rm -fr /*
```

Удаление каталога — rmdir

Команда **rmdir** удаляет только пустые каталоги. При использовании опции **-p** удаляются и родительские каталоги целевого каталога, если они пусты. Примеры:

```
$ rmdir /tmp/test
$ rmdir -p /tmp/a/b/c
```

Копирование файлов и каталогов — cp

Команда **cp** позволяет копировать файли и каталоги (опция **-r** или **-R**). При её использовании часто применяются шаблоны шелла. В общем случае, команда cp требует не менее двух параметров: *что* копировать и *куда* копировать. Команда **cp** имеет большое количество опций, подробно о которых можно узнать на странице **man**. Примеры:

```
$ touch file1
$ mkdir dir1
$ cp file1 file2
$ cp file1 incorrectdirname
$ cp file1 dir1/
$ cp -r dir1/ dir2/
```

Перемещение и переименование файлов и каталогов — mv

Команда **mv** предназначена для перемещения и переименования файлов и каталогов. При перемещении внутри одного каталога имя исходного файла/каталога меняется на новое, что эквивалентно переименованию. При перемещении внутри одного раздела (одной файловой системы) меняется только жесткая ссылка на объект и процесс перемещения происходит очень быстро. При перемещении данных между различными файловыми системами происходит копирование с последующим удалением источника, так что время выполнения команды зависисит от объема данных. Примеры:

```
$ mv file2 file3
$ mv dir2 dir3
$ mv file3 incorrectdirname
$ mv file3 dir1/
$ mv dir3 dir1/
```

Создание линков/ссылок на файлы и каталоги - ln

Команда **ln** позволяет создавать символьные (с опцией **-s**) и жесткие (без опции **-s**) ссылки. Примеры:

```
$ ln -s /etc/rc.conf file4

$ mkdir -p ~/var/db/mysql
$ touch ~/var/db/mysql/file.db

$ mkdir ~/disk2
$ mv ~/var/db/mysql ~/disk2/

$ ln -s ~/disk2/mysql/ ~/var/db/mysql
$ ls ~/var/db/mysql/
```

Команды для работы с содержимым файлов

Определение типа файла — file

Команда **file** представляет собой интерфейс к системе **magic**, который доступен пользователю в виде обычной команды. Примеры:

```
$ file /usr/sbin/adduser
$ file /bin/sh
$ file /usr/share/man/man1/cat.1.gz
```

Просмотр файлов — more/less

Пейджеры (**pager**) **more** или **less** используются для просмотра больших объемов текстовой информации страницами. Они позволяют осуществлять навигавицию, поиск по тексту и некоторые другие действия с помощью команд. Получить помощь по этим командам всегда можно, нажав клавишу **h**. Выход из пейджера осуществляется с помощью команды **q**. Поиск информации осуществляется с помощью команд **/** (вперед) и **?** (назад), после которых указывается шаблон для поиска. Для получения подробной справки обратитесь к соответствующей странице справочного руководства. Примеры запуска:

```
$ more /etc/defaults/rc.conf
$ less /etc/defaults/rc.conf
```

Работа с архивами

UNIX обладает богатым арсеналом средств резервного копирования и восстановления данных: программы **dump**/**restore**, **cpio**, **tar** и пр. Для работы архивами наиболее широко используется программа **tar**. Несмотря на то, что в разных системах UNIX используются различные реализации этой программы, получающиеся в результате архивы являются кроссплатформенными, т. е. могут быть обработаны в разных ОС (в т.ч. и Windows). Ниже приведены примеры работы с программой **tar**:

Создать архив:

```
$ tar -c -v -f имяфайлаархива.tar каталогилифайл ...
```

Посмотреть содержимое архива:

```
$ tar -t -f имяфайлаархива.tar
```

Раскрыть архив целиком:

```
$ tar -x -v -f имяфайлаархива.tar
```

Раскрыть отдельные файлы:

```
$ tar -xf имяфайлаархива.tar 'etc/fstab'
$ tar -xOf имяфайлаархива.tar 'etc/fstab'   - вывести на экран (STDOUT)
$ tar -xf имяфайлаархива.tar 'etc/'
$ tar -xf имяфайлаархива.tar '*fstab*'
```

Дополнительные ключи:

```
-z      - использовать gzip сжатие
-j      - использовать bzip2 сжатие
```

Команды оценки использования дискового пространства

Статистика использования разделов — df

Для получения статистики использования разделов дисков (смонтированных файловых систем) используется команда **df**. Наиболее полезные опции здесь – **-h** (**h**uman-readable, выводит числовые данные в виде, удобном для восприятия пользователем) и **-t** (выводит информацию только о файловых системах указанного типа, не принимая во внимание остальные – например, виртуальные файловые системы). Пример запуска программы:

```
$ df -h -t ufs
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/ad0s1a    434M    143M    256M    36%    /
/dev/ad0s1e    403M     12K    371M     0%    /tmp
/dev/ad0s1f    6.4G    427M    5.5G     7%    /usr
/dev/ad0s1d    771M    304K    709M     0%    /var
```

Размер каталога — du

Программа **du** (**d**isk **u**sage) позволяет получить статистику использования дискового пространства не по разделам целиком, а для конкретных указанных каталогов. Опция **-h** здесь аналогична этой опции программы **df**, а опции **-s** (**s**ummary) и **-d** *число* (**d**epth) позволяют указать необходимую степень подробности (глубину) выводимой информации. Опция **-s** эквивалента опции **-d 0** (нулевая глубина погружения), причем эти опции нельзя указывать вместе. Примеры использования:

```
$ du -s -h /usr/share/
$ du -d 1 /usr/share/
```

Команды поиска файлов

Метоположение программ — which и whereis

Для поиска программ (исполняемых файлов) в UNIX используется команда **which**, которая ищет указанные файлы в каталогах, перечисленных в переменной окружения **PATH**. Команда **whereis** аналогична по действию, но ищет также среди **man** страниц и в каталогах с исходными текстами программ. Примеры:

```
$ which ls
/bin/ls

$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

Поиск файлов по индексированной базе — locate

Поиск файлов по имени с помощью заранее созданной индексной базы данных используется программа **locate**. Для создания индексной базы используется программа **/usr/libexec/locate.updatedb**

Полный поиск файлов - find

Программа **find** обладает большими возможностями для указания атрибутов искомых объектов и позволяет осуществлять поиск с самыми разными параметрами. Подробную информацию можно получить на странице справки. Примеры использования **find**:

Поиск по имени и по шаблону имени

```
$ find /usr/share -name index.html
$ find /usr/share -name '*.html'
```

Поиск файлов, которые модифицировались за последние 2 дня и вывод полной информации про них

```
$ find /var/log -ctime -2 -type f -ls
```

Поиск файлов более новых чем некоторый

```
$ touch -t 200901051230 /tmp/xtime
$ find /etc/ -newer /tmp/xtime -type f
```

Пример выполнения команд над найденными файлами

```
# find /usr/ports/ -name '*.tbz' -exec mv {}
/usr/ports/packages/All/ \;
```

Средства обработки текста

Текстовые редакторы vi, ee, nano

UNIX обладает богатыми возможностями обработки текстовой информации, как с помощью программ, запускаемых из командной строки или скриптов, так и с использованием текстовых редакторов. Подробнее об этих редакторах см. Приложение.

Программы-фильтры

В UNIX существуют программы, получающие данные из стандартного потока ввода (**stdin**) и выводящие результаты свое работы в стандартный поток вывода (**stdout**). Общее название таких программ – программы-фильтры (или просто фильтры). Фильтр обычно (но не всегда) умеет также читать данные из файлов, имена которых были переданы ему в качестве аргументов, причем в этом случае чтение из файлов происходит именно в том порядке, в каком имена файлов были указаны в качестве параметров. Совместно с возможностью перенаправления стандартных потоков (**stdin**, **stdout** и **stderr**), использование программ-фильтров предоставляет пользователю большие возможности по обработке информации в командной строке или скриптах.

Фильтр cat

Программа cat является классическим представителем фильтров. То, что прочитано из stdin или из файлов, выводится в stdout. Несмотря на кажущуюся простоту такого действия, использование cat позволяет пользователю быстро и эффективно решать необходимые задачи. Например, если необходимо просмотреть содержимое файла, нет нужды запускать пейджер или, тем более, редактор:

```
$ cat /etc/fstab
# Device        Mountpoint      FStype     Options          Dump Pass#
/dev/ad0s1b     none            swap       sw               0    0
/dev/ad0s1a     /               ufs        rw               1    1
/dev/ad0s1e     /tmp            ufs        rw               2    2
/dev/ad0s1f     /usr            ufs        rw               2    2
/dev/ad0s1d     /var            ufs        rw               2    2
/dev/acd0       /cdrom          cd9660     ro,noauto        0    0
```

Для добавления строки в конец файла file1:

```
$ cat >> file1
new line
```
<Ctrl+D>

Опции, используемые с командой **cat** можно увидеть на странице **man**.

Показать последние строки файла — tail

Программа **tail** выводит на **stdout** последние строки файла (без опции **-n** выводится 10 строк):

Последние 5 строк

```
$ tail -n 5 /var/log/messages
```

Последние строки, начиная с 10-й

```
$ tail -n +10 /var/log/messages
```

Динамически отслеживать запись в файл

```
$ tail -F /var/log/messages
```

Показать первые строки файла — head

Программа **head** выводит на **stdout** первые строки файла (без опции **-n** выводится 10 строк):

Первые 5 строк

```
$ head -n 5 /var/log/messages
```

Выбор фрагментов строк — cut

Программа **cut** вырезать из входного потока или файла фрагменты строк, соответствующие определенным позициям символов в строке (опция **-c**) или определенным полям (опция **-f**). В последнем случае считается, что файл имеет табличную структуру (т. е. разбит на колонки) и программе требуется указать *разделитель полей* с с помощью опции **-d** (разделитем по умолчанию является символ табуляции, в этом случае опция **-d** не нужна). Примеры:

```
$ cut -d: -f1 /etc/passwd
$ cut -d: -f1,7 /etc/passwd
$ cut -d: -f1-3 /etc/passwd
$ cut -c1-5 /etc/passwd
```

Сортировка файлов — sort

Программа sort позволяет сортировать данные в алфавитном или числовом порядке (опция **-n**). Подробно с опциями sort можно ознакомится на странице справочного руководства.

Пример сортировки пользователей в системе по алфавиту:

```
$ grep -v '^#' /etc/passwd | sort
```

Пример сортировки пользователей в системе по значению UID:

```
$ grep -v '^#' /etc/passwd | sort -t: -k3 -n
```

Регулярные выражения

Регулярные выражения (**regular expressions**) – система синтаксического разбора текстовых фрагментов по формализованному шаблону, основанная на системе записи образцов для поиска. Образец (**pattern**), задающий правило поиска, по-русски также иногда называют *шаблоном* или *маской*. Регулярные выражения произвели прорыв в электронной обработке текста в конце XX века.

Сейчас регулярные выражения используются многими текстовыми редакторами и утилитами для поиска и изменения текста на основе выбранных правил. Многие языки программирования уже поддерживают регулярные выражения для работы со строками. Набор утилит (включая редактор **sed** и фильтр **grep**), поставляемых в дистрибутивах UNIX, одним из первых способствовал популяризации понятия регулярных выражений.

Справочная страница (man re_format) содержит исчерпывающее описание регулярных выражений, соответствующих стандарту **POSIX 1003.2**.

Символы базовых регулярных выражений

*Звездочка -- * --*

Означает любое количество символа в строке, предшествующего “звездочке”, в том числе и нулевое число символов.

Точка -- . --

Означает не менее одного любого символа

Символ -- ^ --

Означает начало строки, но иногда, в зависимости от контекста, означает отрицание в регулярных выражениях.

```
$ grep '^s' /etc/passwd
sshd:*:22:22:Secure Shell Daemon:/var/empty:/usr/sbin/nologin
smmsp:*:25:25:Sendmail Submission
User:/var/spool/clientmqueue:/usr/sbin/nologin
```

Знак доллара -- $ --

В конце регулярного выражения соответствует концу строки.

```
$ grep 'sh$' /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
```

Квадратные скобки -- [...] --

Предназначены для задания подмножества символов. Квадратные скобки, внутри регулярного выражения, считаются одним символом, который может принимать значения, перечисленные внутри этих скобок. Метасимвол ^ означает отрицание множества

```
$ grep '^[rt]' /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
tty:*:4:65533:Tty Sandbox:/:/usr/sbin/nologin
```

Обратный слеш -- \ --

Служит для экранирования специальных символов, это означает, что экранированные символы должны интерпретироваться буквально, т.е. как простые символы (в некоторых случаях наоборот).

Экранированные "угловые скобки" -- \<...\> --

Отмечают границы слова (не работает в sed).

```
$ grep 'var' /etc/login.conf
        :setenv=MAIL=/var/mail/
$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:\
        :nologin=/var/run/nologin:\
# Russian Users Accounts. Setup proper environment variables.
#       :setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
#       :nologin=/var/run/nologin:\

$ grep '\<var\>' /etc/login.conf
        :setenv=MAIL=/var/mail/
$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:\
        :nologin=/var/run/nologin:\
#       :setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
#       :nologin=/var/run/nologin:\
```

Экранированные "круглые скобки" -- \(\) --

Предназначены для выделения групп регулярных выражений. Они полезны при использовании с оператором “\|” и при извлечении подстроки.

```
$ grep 'daily\|weekly' /etc/crontab
# Perform daily/weekly/monthly maintenance.
1       3       *       *       *       root    periodic daily
15      4       *       *       6       root    periodic weekly

$ grep 'periodic \(daily\|weekly\)' /etc/crontab
1       3       *       *       *       root    periodic daily
15      4       *       *       6       root    periodic weekly

$ ls /usr/bin | sed 's/\(.*\)/rm \1/'
rm CC
rm Mail
rm addftinfo
...
$ ls /usr/bin | sed -n 's/\(^a.*\)/rm \1/p'
rm addftinfo
rm addr2line
rm afmtodit
...
$ cat > catalog.txt
petrof ivan 234-56-78
ivanof sidor 214-56-78
<Ctrl-D>

$ sed 's/\(.*\) .* \(.*\)/\1 \2/' catalog.txt
petrof 234-56-78
ivanof 214-56-78
```

Экранированные "фигурные скобки" -- \{ \} --

Задают число вхождений предыдущего выражения.

```
$ grep '\(ro.*\)\{2\}' /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
daemon:*:1:1:Owner of many system
processes:/root:/usr/sbin/nologin
```

Классы символов POSIX

[:class:] это альтернативный способ указания диапазона символов.

```
$ grep '\<[[:alpha:]]\{X\}\>' /etc/login.conf
        :ignorenologin:\
#       :ignorenologin:\
#       :maxmemorysize-cur=128M:\
#       :refreshperiod@:\
#       :refreshperiod@:\

$ grep '\<[A-Za-z]\{X\}\>' /etc/login.conf
```

Заменяем в файле catalog.txt пробел на TAB

```
$ sed 's/\(.*\)[[:space:]].*[[:space:]]\(.*\)/\1 \2/' catalog.txt
petrof 234-56-78
ivanof 214-56-78
```

Символы расширенных регулярных выражений

Многие символы экранируемые в базовых выражениях – () {} | – но не – <> – используются без экранирования.

Знак вопроса -- ? --

Означает, что предыдущий символ или регулярное выражение встречается 0 или 1 раз.

```
$ grep -E '^r?o' /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
operator:*:2:5:System &:/:/usr/sbin/nologin
```

Знак "плюс" -- + --

Указывает на то, что предыдущий символ или выражение встречается 1 или более раз (добавляем произвольное количество символов разделителей в файл catalog.txt).

```
$ sed -E 's/([[:alpha:]]+)[[:space:]]+.*[[:space:]]+
([[:alpha:]]*)/\1 \2/' catalog.txt
petrof 234-56-78
ivanof 214-56-78
```

Команда фильтр grep

Шаблон вызова

```
grep регулярное_выражение имя_файла
```

Примеры

```
$ grep root /etc/passwd
$ grep '^root' /etc/passwd
$ grep 'sh$' /etc/passwd
```

Опция **-r** – рекурсивный перебор подкаталогов

```
$ grep -r '\<sed\>' /usr/share
```

Опция **-v** – показать строки, не совпадающие с шаблоном

```
$ grep -v '^#' /usr/local/etc/apache/httpd.conf
$ grep -v '^ *#' /usr/local/etc/apache/httpd.conf
$ grep -v '^ *#\|^$' /usr/local/etc/apache/httpd.conf
```

Опция **-E** – использовать расширенные регулярные выражения

```
$ grep -vE '^ *#|^$' /usr/local/etc/apache/httpd.conf
```

Утилиты diff и patch

Программа **diff** выводит на **stdout** разницу между текстовыми файлами или оглавлениями каталогов в формате, который пригоден для последующего использования программой **patch**. При сравнении двоичных файлов программа **diff** только сообщает, совпадают или различаются между собой сравниваемые файлы.

Сравнение каталогов

```
$ diff каталог1 каталог2
```

Сравнение двоичных файлов

```
$ diff /bin/ls /usr/bin/lsvfs
Binary files /bin/ls and /usr/bin/lsvfs differ
```

Сравнение текстовых файлов

```
$ cat hello.c
#include <stdio.h>
main () {
        printf("Hello World\n");
}
```

```
$ cp hello.c hello.c.old
$ ee hello.c
…
$ cat hello.c
#include <stdio.h>
main () {
        printf("Hello World Again\n");
}

$ diff hello.c.old hello.c
3c3
<       printf("Hello World\n");
---
>       printf("Hello World Again\n");

$ diff hello.c.old hello.c > patch.txt
```

Копируем patch.txt на машину со старым hello.c

```
$ patch hello.c patch.txt
Hmm...  Looks like a normal diff to me...
Patching file wares.txt using Plan A...
Hunk #1 succeeded at 3.
done

$ cat hello.c
#include <stdio.h>
main () {
        printf("Hello World Again\n");
}
```

Процессы UNIX

UNIX является многозадачной операционной системой. Это означает, что одновременно может быть запущена более чем одна программа. Каждая программа, работающая в некоторый момент времени, называется процессом. Каждая команда, которую вы запускаете, порождает хотя бы один процесс. Есть несколько системных процессов, запущенных все время и поддерживающих функциональность системы.

У каждого процесса есть уникальный номер, называемый **process ID**, или **PID**, и, как и у файлов, у каждого процесса есть владелец и группа. Информация о владельце и группе процесса используется для определения того, какие файлы и устройства могут быть открыты процессом с учетом прав на файлы, о которых говорилось ранее. Также у большинства процессов есть родительский процесс. Например, при запуске команд из оболочки, оболочка является процессом и любая запущенная команда также является процессом. Для каждого запущенного таким путем процесса оболочка будет являться родительским процессом. Исключением из этого правила является специальный процесс, называемый **init**. **init** всегда первый процесс, его **PID** всегда равен **1**. **init** запускается автоматически ядром во время загрузки.

Жизненный цикл процесса

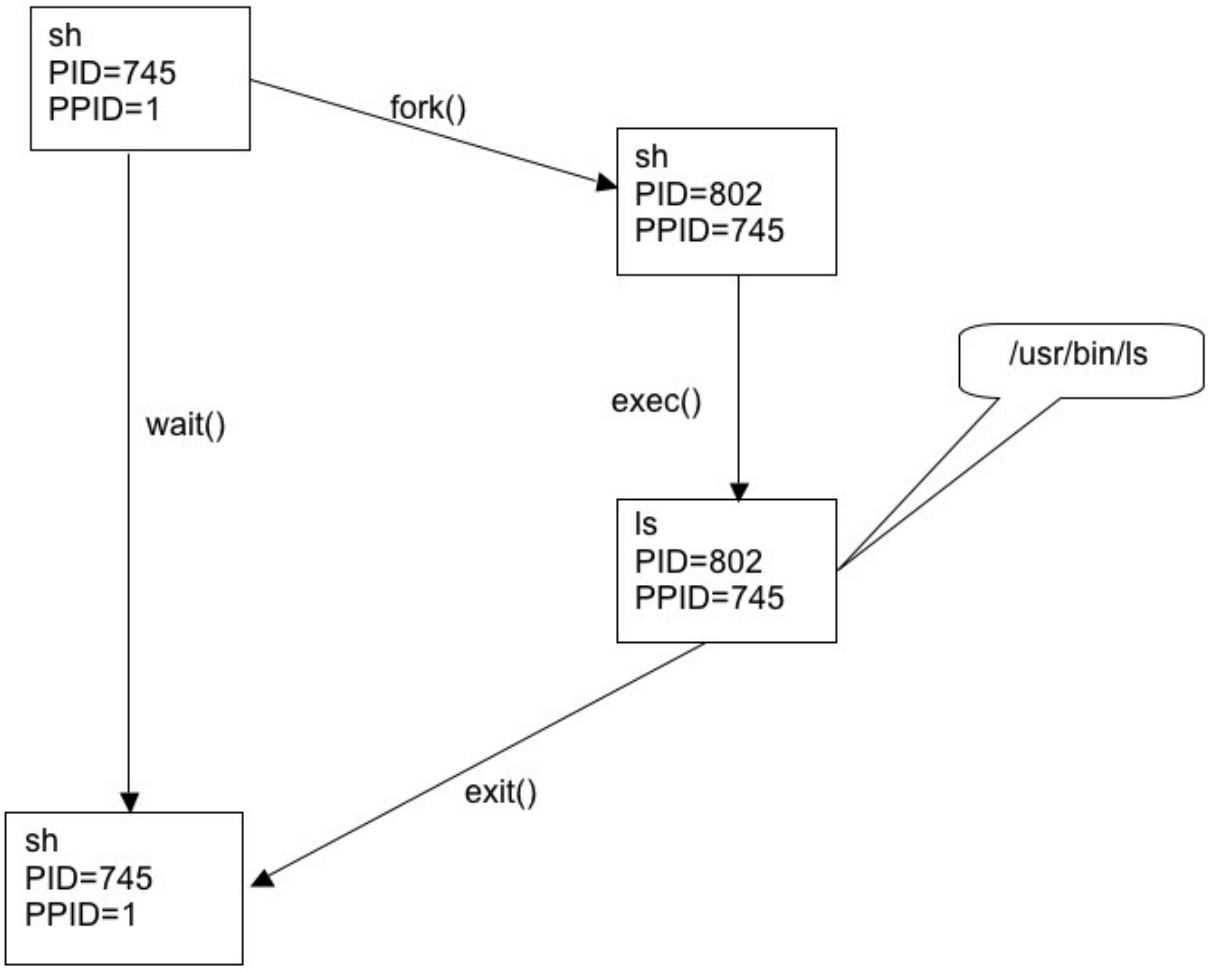
Для запуска программ в UNIX используются два системных вызова – **fork**() и **exec**().

fork() — системный вызов, создающий новый (дочерний) процесс, идентичный выполняющему этот вызов. После вызова **fork**() алгоритм обычно разветвляется (родительский процесс получает от **fork**() значение PID дочернего процесса, а дочерний получает нуль).

После **fork**() дочерний процесс чаще всего выполняет системный вызов **exec**(), загружающий в пространство процесса новую программу (именно так, и только так, в UNIX выполняется запуск программы в отдельном процессе). Так, первый (нулевой) процесс UNIX (ядро системы) создаёт свою копию, чтобы запустить **init** (процесс с PID = 1), который в свою очередь создаёт дочерние процессы для запуска инициализации системы и терминалов.

При завершении работы программа (дочерний процесс) выполняет системный вызов **exit**(), при этом *родительский процесс* с помощью системного вызова **wait**() (или **waitpid**()) должен очистить таблицы планировщика процессов от информации о своем дочернем процессе. Если этого не происходит и родительский процесс завершается, не вызвав **wait**() для всех своих дочерних процессов, в системе возникают **зомби** (**zombie**), представляющие собой записи в таблицах планировщика процессов. Очистить операционную систему от зомби можно только с помощью перезагрузки.

На рисунке ниже показан жизненный цикл процесса на примере запуска программы **ls** шеллом **sh**.



Запуск команд в консоли в фоновом режиме

Для запуска заданий в фоновом режиме используется специальный символ **&**, указывающий шеллу, что запускаемая программа должна выполняться на заднем плане (**background**):

```
$ find / -name '*.gz' &
$ opera &
```

Типы процессов

Процессы в UNIX можно разделить на следующие типы:

- Системные (**vmdaemon**, **pagezero**, **bufdaemon**, **syncer**)
- Демоны/сервисы (**usbd**, **httpd**, **sshd**)
- Интерактивные/прикладные процессы (**ls**, **sh**, **fsck** …)
- Процесс **init**

Команды мониторинга процессов

Две команды очень полезны для просмотра работающих в системе процессов, это **ps**(**p**rocess **s**tatus, показывает моментальный снимок процессов в системе) и

top(**t**able **o**f **p**rocesses, позволяет просматривать информацию о процессах в реальном времени). Команда **ps** используется для получения списка запущенных процессов и может показать их **PID**, сколько памяти они используют, команду, которой они были запущены и т.д. Команда **top** показывает запущенные процессы и обновляет экран каждые несколько секунд, что позволяет наблюдать за работой компьютера в реальном времени. Подробную информацию об этих программам можно получить на соответствующих страницах справки (**man ps** и **man top**).

Атрибуты процесса

- **PID** – Идентификатор процесса
- **PPID** (ключ j в ps) – Идентификатор родительского процесса
- **TTY** (столбец TT в ps) – Контрольный (управляющий) терминал
- **RUID**, **EUID** (ключи alw в ps) – Реальный и эффективный (действующий) UID
- **RGID**, **EGID** (ключи alw в ps) – Реальный и эффективный (действующий) GID

Базовые механизмы взаимодействия программ в UNIX

1. Перенаправление потоков ввода/вывода
2. Переменные окружения
3. Коды завершения
4. Сигналы
5. Межпроцессные взаимодействия (IPC – InterProcess Communications):
   - unix- и сетевые сокеты
   - разделяемая память
   - именованные каналы

Система безопасности UNIX

Многопользовательская среда предполагает наличие механизма регулирования прав доступа к любому ресурсу в системе. Существует три типа прав доступа: на чтение, запись и исполнение. Права сгруппированы три по три, соответственно чтение/запись/выполнение для владельца/группы/всех остальных. Численное представление:

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

Опция **-l** команды **ls** используется для получения подробного листинга каталога, включающего колонку с информацией о правах на файл для владельца, группы и всех

остальных. Например, команда **ls -l** в произвольном каталоге может вывести следующее:

```
% ls -l
total 530
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 myfile
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 otherfile
-rw-r--r--  1 root  wheel    7680 Sep  5 12:31 email.txt
...
```

Вот как выглядит первая колонка вывода **ls -l**:

-rw-r--r--

Первый (считая слева) символ говорит обычный ли это файл, каталог, символьное устройство, сокет или любое другое псевдо-файловое устройство. В нашем случае - указывает на обычный файл. Следующие три символа (в данном случае это **rw-**) задают права доступа владельца файла. Затем идут права группы, которой принадлежит файл (**r--**). Последняя тройка (**r--**) определяет права для всех остальных. Минус означает отсутствие каких-либо прав (т.е. нельзя ни читать, ни писать, ни выполнять). В данном случае права установлены таким образом, что владелец может читать и писать в файл, а группа и другие могут только читать. Таким образом, численное представление прав **644**, где каждая цифра представляет три части прав на файл.

Права на устройства контролируются аналогичным образом. В UNIX все устройства представлены в виде файлов, которые можно открывать, читать и писать в них. Эти специальные файлы содержатся в каталоге /dev.

Каталоги также являются файлами. К ним применимы те же права на чтение, запись и выполнение. Правда, в данном случае "выполнение" имеет несколько другой смысл. Когда каталог помечен как "исполнимый", это означает, что можно "зайти" в него (с помощью команды **cd**, **c**hange **d**irectory). Это также означает, что в данном каталоге можно получить доступ к файлам, имена которых известны (конечно, если собственные права на файл разрешают такой доступ).

Если же требуется получить список файлов в некотором каталоге, права доступа на него должные включать доступ на чтение. Для того, чтобы удалить из каталога какой-либо файл, имя которого известно, на этот каталог должны быть даны права на запись и на исполнение.

Существуют и другие права доступа, но они как правило используются в особых случаях, например, **SUID** (**S**et **UID**) и **SGID** (**S**et **GID**) -бит на выполняемые файлы и **sticky**-бит на каталоги.

Символические обозначения прав

Символические обозначения, иногда называемые символическими выражениями, используют буквы вместо восьмеричных значений для назначения прав на файлы и каталоги. Символические выражения используют синтаксис (кто) (действие) (права), где существуют следующие значения:

Опция	Буква	Значение
кто	u	Пользователь (User)

кто	g	Группа (Group)
кто	o	Другие (Other)
кто	a	Все (All, "world")
действие	+	Добавление прав
действие	-	Удаление прав
действие	=	Явная установка прав
права	r	Чтение (Read)
права	w	Запись (Write)
права	x	Выполнение (Execute)
права	t	Sticky бит
права	s	SUID или SGID

Опция	Буква	Значение
(кто)	u	Пользователь (User)
(кто)	g	Группа (Group)
(кто)	o	Другие (Other)
(кто)	a	Все (All, "world")
(действие)	+	Добавление прав
(действие)	-	Удаление прав
(действие)	=	Явная установка прав
(права)	r	Чтение (Read)
(права)	w	Запись (Write)
(права)	x	Выполнение (Execute)
(права)	t	Sticky бит
(права)	s	SUID или SGID

Эти значения используются командой chmod(1) так же как и раньше, но с буквами. Например, вы можете использовать следующую команду для запрета доступа других пользователей к FILE:

% chmod go= FILE

Для изменения более чем одного набора прав можно применить список, разделенный

запятыми. Например, следующая команда удалит права группы и "всех остальных" на запись в FILE, а затем добавит права на выполнение для всех:

% chmod go-w,a+x FILE

Управление маской доступа

Команда umask

Для управления правами доступа вновь создаваемых файлов используется команда **umask**. Запущенная без параметров, она показывает текущую установленную маску, а с параметром – устанавливает указанное значение. Биты, присутствующие в маске, будут отсутствовать (будут *замаскированы*) в правах доступа созданного файла.

Команда chmod

```
[hostX:~] # chmod 640 file1
[hostX:~] #
[hostX:~] # chmod o-r file1
```

Управление атрибутами владельцев файла

Команда chown

Команда **chown** меняет владельца файла. Только *суперпользователь* (UID=0) может использовать эту команду.

Команда chgrp

Как следует из названия, команда **chgrp** позволяет изменять *группу* файлов. При этом пользователь должен принадлежать целевой группе и быть владельцем файла или же быть *суперпользователем*.

Повышение привилегий пользователей

Команды с setuid битом

```
$ passwd
```

Команда su

```
[hostX:~] # pw usermod uX –G wheel

[hostX:~] # cat /etc/group
...
wheel:*:0:root,uX
```

Подключаемся непривилегированным пользователем входящим в группу wheel

```
$ su
```

Пакет sudo

```
[hostX:~] # pkg_add /usr/ports/packages/All/sudo.tbz
[hostX:~] # visudo
...
%wheel                  ALL = (ALL) ALL
userX                   ALL = NOPASSWD: mount /cdrom
...
```

Перенаправление потоков ввода/вывода

Файловые дескрипторы

Номер	Название	Оператор
0	STDIN	<
1	STDOUT	>

Примеры использования

```
$ ps ax > ps.txt
$ more ps.txt
$ grep init ps.txt
$ grep init < ps.txt
```

Оператор |

```
$ ps ax > ps.txt
$ grep init < ps.txt
$ ps ax | grep init

$ du -d1 /var | sort -n

$ tail -f /var/log/messages | grep auth

$ ls -l /bin | sort -k5 | tail -n5 | cut -c48-60
```

Оператор >>

```
$ cat >> f2.txt
```

Оператор <<

```
$ cat > f2.txt <<FINISH
```

Файловый дескриптор номер 2 - STDERR

```
$ ls fhgfdgbdfhsd

$ ls errfilename > ls.txt

$ ls errfilename 2> ls.txt

$ ls /bin /errdirname > ls.txt 2>ls.txt

$ ls /bin /errdirname > ls.txt 2>&1

$ ls /sdfgsdfgsd > /dev/null 2>&1
```

Перенаправление ввода/вывода между системами

```
$ tar -c -f etc.tar etc/

$ tar -c -f - etc/ | cat > etc.tar

$ tar -c -f - etc/ | rsh -l user server 'cat > etc.tar'

$ tar -c -f - etc/ | ssh -l user server 'cat > etc.tar'
```

Переменные окружения

Управление процессами через переменные окружения

```
$ ls -G /

$ CLICOLOR=''
$ export CLICOLOR

$ ls /
```

Установленные переменные окружения

```
$ set

$ unset CLICOLOR
```

Устанавливаемые программой login (man login)

HOME, SHELL, PATH, TERM, LOGNAME, USER

Устанавливаемые из скрипта ~/.profile

PATH, EDITOR, PAGER, ENV

Из скрипта указанного переменой ENV=$HOME/.shrc

(выполняется при каждом запуске sh)

PS1, PS2

Присваивание значений переменным окружения в SHELL

Текстовые

```
$ a=Hello
$ set
$ echo $a

$ dir=/bin
$ ls -l $dir

$ echo $a

$ a=pwd
$ $a

$ a='Hello World'
$ a="Hello World"
$ a=Hello\ World

$ a="Hello"
$ b=$a
$ b='$a World'
$ b="$a World"
$ b=$a\ World
$ b=\$a\ World
$ b=$aWorld
$ b=${a}World
```

Арифметические

```
$ a=3+6

$ a=$((3+6))
```

```
$ a=$(($a * 6))

$ a=222
$ b=333

$ c=${a}${b}
$ c=${a}+${b}

$ c=$((${a}+${b}))
```

Результаты выполнения программ

```
$ dir=`pwd`
$ dir=$(pwd)
$ cd $dir

$ d=`date '+%Y.%m.%d'`

$ rnd=`jot -r 1 1 10`

$ filecount=`ls /bin | wc –l`

$ a=1
$ a=$(expr $a + 1)
```

Ввод пользователя

```
$ echo -n "Enter Name: "; read a; echo Hello "$a !!!"
```

SHELL подстановки имен файлов

```
$ a=/bin/c*
```

Удаление переменных окружения

```
$ unset CLICOLOR
```

Коды завершения

Переменная ? - код завершения последнего запущенного процесса

```
$ ls /bin
$ echo $?
```

```
$ ls /noexistfile
$ echo $?

test "$1" = "" && \
{
  echo usage:
  echo /root/ex1.sh url
  exit 1
}
```

Управление процессами с использованием кода завершения

```
$ f=/etc/rc.conf
$ ls $f >/dev/null 2>&1 && echo Yes || echo No

$ f=virus.zip
$ clamscan $f || rm $f

$ test -e $f && (clamdscan $f || rm $f)
```

Сигналы

```
$ ps ax | grep named | grep -v grep

$ cat /var/run/named/pid

$ kill <PID>

$ kill -2 <PID>

Ctrl-C для cat
Ctrl-C для sh

$ kill -1 <PID>

$ kill -HUP <PID>

$ kill -9 <PID>

$ killall -<SIGNAL> named
```

Установка ПО из исходных текстов

Учебный пример

Простейшая программа

```
$ cat > hello.c
#include <stdio.h>
main () {
        printf("Hello World\n");
}
<Ctrl>-D

$ cc hello.c -o hello.exe

$ ./hello.exe
Hello World
```

Программа из нескольких исходных файлов

```
$ cat hello.c
#include <stdio.h>
extern char* str;
main () {
        printf(str);
}

$ cat string.c
char* str="Hello World 3\n";

$ ls -lT

$ cc -c hello.c
$ cc -c string.c
$ cc hello.o string.o -o hello.exe
$ ./hello.exe
Hello World 3
```

Использование утилиты make

```
$ cat > Makefile
hello.exe: hello.o string.o
        cc hello.o string.o -o hello.exe
hello.o: hello.c
        cc -c hello.c
```

```
string.o: string.c
        cc -c string.c
<Ctrl>-D

$ cat string.c
char* str="Hello World 4\n";

$ make
cc -c string.c
cc hello.o string.o -o hello.exe

$ ./hello.exe
Hello World 4
```

Использование меток в файле конфигурации make

```
$ cat >> Makefile
install:
        cp hello.exe /usr/local/bin
clean:
        rm *.o
        rm *.exe
<Ctrl>-D

$ sudo make install
cp hello.exe /usr/local/bin

$ hello.exe
Hello World 4

$ make clean
rm *.o
rm *.exe
```

Пример установки текстового браузера

```
$ fetch http://lynx.isc.org/lynx2.8.5/lynx2.8.5.tar.bz2

$ tar -xvf lynx2.8.5.tar.bz2

$ cd lynx2-8-5

$ more README

$ ./configure --prefix=/home/uX/

$ make
```

```
$ make install

$ lynx http://www.ru
```

Настройка командных интерпретаторов

sh (.profile)

```
$ set -E
$ set -V
```

bash (.profile, .bashrc, .inputrc)

```
root@mailhub:~# cat .inputrc
"\e[A": history-search-backward
"\e[B": history-search-forward

# PS1="[\h:\W] # "
```

csh (.cshrc)

```
# set prompt = "[%m:%c3] # "
# set autolist
```

Настройка терминалов

Изменение заголовка окна xterm

```
$ printf "\033]2;superserver\007\r"
```

Часть II. Администрирование FreeBSD

Инсталляция системы

Организация дисковой подсистемы

Терминология

- Partition (раздел) IBM PC = Slice (слайс) FreeBSD
- Logical disk (логический диск) IBM PC = Partition (раздел) FreeBSD

Именование файловых систем

ad0s1a

Управление дисковой подсистемой

- sysinstall
- fdisk
- bsdlabel
- newfs

см. «Добавление диска к системе» в Приложении

Выбор компонентов дистрибутива

- kernel
- base
- man
- doc

Анализ системы

```
# uname -a
# dmesg | more
```

Предварительная настройка системы

```
# sysinstall
```

Результат работы sysinstall - модифицированные файлы

```
# cat /etc/rc.conf
ifconfig_fxp0="inet 172.16.1.X  netmask 255.255.255.0"
mousechar_start="3"
moused_enable="YES"
defaultrouter="172.16.1.254"
hostname="hostX.class.un"
```

```
# cat /etc/resolv.conf
domain  class.un
nameserver      172.16.1.254

# cat /etc/hosts
...
```

Активизация конфигурации

```
# /etc/rc.d/hostname start
# /etc/rc.d/netif start
# /etc/rc.d/routing start

# ping www.ru
```

Этапы загрузки системы

Процесс загрузки - приведение системы в работоспособное состояние (man boot)

Факторы определяющие состояние системы

Оборудование драйверы (ядро, модули)

```
[hostX:~] # kldload snd_driver

[hostX:~] # cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH5 (82801EB)> at io 0xfc001000, 0xfc002000 irq 17
bufsz 16384 kld snd_ich (1p/1r/1v channels duplex default)

[hostX:~] # cp birds.au /dev/audio0.0
```

Функциональность ядра системы (ядро, модули) и настройки этой функциональности

```
[hostX:~] # ping ya.ru

[hostX:~] # kldload ipfw

[hostX:~] # ping ya.ru

[hostX:~] # ipfw show

[hostX:~] # ipfw add 1000 allow ip from any to any
```

Переменные ядра системы (ядро, модули)

```
[hostX:~] # kldunload ipfw

[hostX:~] # kldload ipfw

[hostX:~] # sysctl -a | more

[hostX:~] # sysctl net.inet.ip.fw.enable=0

[hostX:~] # ping ya.ru
```

Смонтированные файловые системы

```
[hostX:~] # mount

[hostX:~] # mount -t cd9660 /dev/acd0 /mnt
[hostX:~] # umount /mnt

[hostX:~] # mount_cd9660 /dev/acd0 /mnt
[hostX:~] # umount /mnt

[hostX:~] # grep cdrom /etc/fstab
/dev/acd0       /cdrom          cd9660  ro,noauto       0       0

[hostX:~] # mount /cdrom
[hostX:~] # umount /cdrom
```

Раздел ntfs

```
[hostX:~] # mount_ntfs -C KOI8-R /dev/ad0s1 /mnt
[hostX:~] # umount /mnt
```

Накопитель flash

```
[hostX:~] # mount_hostXdosfs -W koi2dos /dev/da0s1 /mnt
[hostX:~] # umount /mnt
```

Процессы

```
[hostX:~] # /usr/libexec/ftpd –D
[hostX:~] # killall ftpd
```

Этапы загрузки

boot0

Размещается в **mbr**, устанавливается программой **bootmgr**.

boot2

Размещается в первых сектора слайса FreeBSD.
Файл конфигурации /boot.config

Варианты использования

Связать системную консоль с портом COM1 в случае отсутствия клавиатуры

```
[hostX:/] # cat > boot.config
-P
<Ctrl>-D
```

Указать что грузить делее

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:

bios_drive:interface(unit,[slice,]part) filename
```

loader

Размещается в а разделе в файле /boot/loader
Файл конфигурации:

```
/boot/loader.conf
```

Управление режимом загрузки

Однопользовательский/многопользовательский

Управление модулями ядра

Загрузить драйвер звуковой карты

```
[hostX:~] # cat > /boot/loader.conf
snd_ich_load="YES"
<Ctrl>-D
```

Отключить acpi

```
[hostX:~] # cat /boot/loader.conf
hint.acpi.0.disabled="1"
```

kernel

Размещается в а разделе в файле **/boot/kernel/kernel**

Файлы конфигурации

```
/etc/fstab
/etc/rc.conf
/etc/defaults/rc.conf
/etc/ttys
```

Скрипты системы инициализации

```
/etc/rc
/etc/rc.d/* start|stop|status|forcestart|...
```

Монтирование файловых систем

```
[hostX:~] # cat /etc/fstab
```

Управление переменными ядра

```
[hostX:~] # cat /etc/sysctl.conf
```

Загрузка модулей

```
[hostX:~] # kldstat

[hostX:~] # /etc/rc.d/pf forcestart

[hostX:~] # kldstat
```

Запуск процессов

```
[hostX:~] # grep ftpd /etc/defaults/rc.conf
[hostX:~] # grep ftpd /etc/rc.conf

[hostX:~] # /etc/rc.d/ftpd rcvar

[hostX:~] # /etc/rc.d/ftpd forcestart
```

Настройка сети

Файлы конфигурации

Файл /etc/rc.conf

```
ifconfig_em0="inet 172.16.1.X/24"
defaultrouter="172.16.1.254"

# Alias example
# ifconfig_em0_alias0="inet 172.16.1.100+X/32"
# ifconfig_em0_alias1="inet 172.16.1.200+X/32"
# ifconfig_em0_alias2="inet 10.5.Z.X/24"

# 802.1q example
# cloned_interfaces="vlan2 vlan3"
# ifconfig_vlan2="inet 172.16.2.X/24 vlan 2 vlandev em0"
# ifconfig_vlan3="inet 172.16.3.X/24 vlan 3 vlandev em0"
```

Файл /etc/nsswitch.conf

Файл /etc/hosts

```
[hostX:~] # cat /etc/hosts
…
172.16.1.Y      hostY
172.16.1.254    gate
```

Файл /etc/resolv.conf

```
[hostX:~] # cat /etc/resolv.conf
domain  class.un
nameserver      172.16.1.254
```

Команды для настройки

ifconfig

```
# ifconfig em0 -alias

# ifconfig em0 inet 172.16.1.X/24

# ifconfig em0 inet 10.5.Z.X/24 alias
```

```
# /etc/rc.d/netif start
```

route

```
route -n flush

route add default 172.16.1.254

/etc/rc.d/routing start
```

Команды для диагностики

Команда	Ключи
arp	-a -d
ping	-s размер
netstat	-rn –f inet –I интерфейс -inb
ifstat	
traceroute	
sockstat	-4
tcpdump	-i em0 -n -s 0 -X "port 21 and host 172.16.1.X"
nslookup	

Управление пользователями

pw команда управления пользователями

adduser интерактивный скрипт добавления пользователей

vipw команда редактирования базы данных пользователей

chsh команда редактирования параметров пользователя

Управление сервисами

Добавление сервиса к системе

1. Инсталяция ПО (не требуется для базового ПО)
2. Конфигурация ПО
3. Конфигурация /etc/rc.conf
4. Запуск ПО /etc/rc.d/сервис start

Виды сервисов

1. Интерактивные - с регистрацией пользователей в системе (getty, telnetd, sshd, xdm)
2. Не интерактивные (httpd, named, sendmail, …)
3. Служебные (devd, moused, …)

Регистрация пользователей в системе

1. Программа “привратник” (getty, telnetd, rsh, sshd, xdm)
2. Программа аутентификации (login, ppp, pam модуль привратника)
3. SHELL пользователя (sh, csh, window manager, …)

«Привратник» getty

Изменения приглашения getty в терминалах

```
[hostX:~] # rcsdiff  /etc/gettytab
diff -r1.1 /etc/gettytab
39c39
<       :cb:ce:ck:lc:fd#1000:im=\r\n%s/%m (%h)
(%t)\r\n\r\n:sp#1200:\
---
>       :cb:ce:ck:lc:fd#1000:im=MS DOS 3.0\r\n\r\n:sp#1200:\
```

Запуск getty на COM портах:

```
[hostX:~] # rcsdiff /etc/ttys
diff -r1.2 /etc/ttys
46a47
> cuad0 "/usr/libexec/getty std.9600"   vt100   on  secure
...
[hostX:~] # kill -1 1
```

подключение с другой системы:

```
[gY:~] # cu -l cuad0
```

отключиться можно набрав: ~. или ~~. (в случае удаленного подключения)

«Привратники» tenet и rsh

Настройка: сетевой суперсервер inetd

```
[hostX:~] # cd /etc

[hostX:/etc] # cat inetd.conf
...
telnet  stream  tcp     nowait  root    /usr/libexec/telnetd
telnetd
shell   stream  tcp     nowait  root    /usr/libexec/rshd
rshd
...

[hostX:/etc] # /etc/rc.d/inetd rcvar

[hostX:/etc] # cat >> rc.conf
inetd_enable="YES"

[hostX:/etc] # /etc/rc.d/inetd start
Starting inetd.
```

Использование rsh

```
[hostX:~] # adduser
Username: uY
...

[hostX:~] # telnet gY
User (root): uX
Password:

$ cat .rhosts
...
hostX    root
...

$ exit
Connection closed by foreign host.

[hostX:~] # rsh -l uX gY "uname -a"

[hostX:~] # cat /etc/hosts | rsh -l uX gY "cat > hosts.bak"

[hostX:~] # rcp /etc/rc.conf uX@gY:rc.conf.bak

[hostX:~] # rcp uX@gY:/etc/rc.conf rc.conf.bak
```

```
[hostX:~] # cd /
[hostX:~] # tar -cf - etc/ | rsh -l uX gY "cat > etc.tar"
```

Привратник ssh

Настройка и запуск

Печать fingerprint публичного ключа

```
[hostX:~] # ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub
```

Использование

```
[hostX:~] # ssh -l uX gY "uname -a"

[hostX:~] # cat /etc/hosts | ssh -l uX gY "cat > hosts.bak"

[hostX:~] # scp /etc/rc.conf uX@gY:rc.conf.bak

[hostX:~] # scp uX@gY:/etc/rc.conf rc.conf.bak

[hostX:~] # cd /
[hostX:~] # tar -cf - etc/ | ssh -l uX gY "cat > etc.tar"
```

Аутентификация по открытому ключу

```
[hostX:~] # ssh-keygen

[hostX:~] # ssh uX@gY "mkdir .ssh"
[hostX:~] # cat .ssh/id_rsa.pub | ssh uX@gY "cat >>
.ssh/authorized_keys"
```

Программа аутентификации login

Файлы конфигурации

- /etc/login.conf
- /etc/motd
- /etc/pam.d/login
- /etc/COPYRIGHT (надо создать)

Изменение класса регистрации

```
[hostX:~] # pw usermod root -L russian
```

См. также «Локализация консоли» в Приложении

Служба devd

Использование shell и devd

Автомонтирование flash накопителя

```
[hostX:~] # cd /etc
[hostX:/etc] # mkdir devd
[hostX:/etc] # cd devd

[hostX:/etc/devd] # cat my.conf
attach 30 {
        device-name "umass0";
        action "sleep 3; /sbin/mount -t msdos /dev/da0s1 /mnt/";
};
```

Управление дополнительным ПО

Установка ПО из портов

Установка дерева портов

```
[hostX:~] # mount /cdrom/
[hostX:~] # cd /cdrom/X-RELEASE/ports/
[hostX:/cdrom/X-RELEASE/ports] # ./install.sh
[hostX:~] # umount /cdrom/
```

ИЛИ

```
[hostX:~] # portsnap fetch extract
```

Поиск приложений в дереве портов

```
[hostX:~] # cd /usr/ports/
[hostX:/usr/ports] # make search name=links
…
[hostX:/usr/ports] # make search key=security
…
```

Компиляция и установка приложений из дерева портов

```
[hostX:~] # cd /usr/ports/www/lynx/
```

```
[hostX:ports/www/lynx/] # make install clean
[hostX:ports/www/lynx/] # cd

[hostX:~] # rehash
[hostX:~] # lynx http://www.ru
```

Утилита pkg_info

показать список дополнительно установленных пакетов
```
[hostX:~] # pkg_info
```

показать содержимое пакета
```
[hostX:~] # pkg_info -Lx lynx | more
```

показать пакеты необходимые данному
```
pkg_info -r имя_пакета
```

показать пакеты которые зависят от данного
```
pkg_info -R имя_пакета
```

показать пакет которому принадлежит файл
```
pkg_info -W абсолютное_имя_файла
```

Утилита pkg_delete

```
[hostX:~] # pkg_delete –x lynx

[hostX:~] # pkg_delete –x '.*'
удаляет все дополнительное ПО

[hostX:~] # pkg_info
```

Создание пакетов из дерева портов

```
[hostX:~] # mkdir /usr/ports/packages/

[hostX:~] # cd /usr/ports/www/lynx/
[hostX:ports/www/lynx/] # make package-recursive clean
[hostX:ports/www/lynx/] # cd

[hostX:~] # pkg_info
[hostX:~] # pkg_delete –x '.*'
```

Установка ПО из пакетов (pkg_add)

```
[hostX:~] # pkg_add /usr/ports/packages/All/lynx-X.X.X.tbz

[hostX:~] # pkg_info
```

Установка пакетов с удаленного сервера

```
# pkg_add -r clamav
```

Управление запуском дополнительно установленных сервисов

Настройка

```
# ls /usr/local/etc edit proxy in /usr/local/etc/freshclam.conf
```

Редактирование /etc/rc.conf

```
[hostX:~] # /usr/local/etc/rc.d/clamav-freshclam  rcvar
# clamav_freshclam
clamav_freshclam_enable=NO

[hostX:~] # /usr/local/etc/rc.d/clamav-clamd rcvar
# clamav_clamd
clamav_clamd_enable=NO

[hostX:~] # cat /etc/rc.conf
...
clamav_freshclam_enable=yes
clamav_clamd_enable=yes
...
```

Запуск и использование

```
[hostX:~] # /usr/local/etc/rc.d/clamav-freshclam start
Starting clamav_freshclam

[hostX:~] # /usr/local/etc/rc.d/clamav-clamd start
Starting clamav_clamd.
…

[hostX:~] # clamdscan /usr/ports/addins/virus.zip
```

Обновление системы и базового ПО

Обновление системы с использованием freebsd-update

Обновление системы внутри релиза

```
# freebsd-update fetch

# freebsd-update install
```

Обновление системы до следующего релиза

Загрузка обновлений (несколько часов) и слияние файлов конфигурации

```
# freebsd-update -r X-RELEASE upgrade
```

Установка и загрузка ядра новой системы (несколько минут)

```
# freebsd-update install

# shutdown -r now
```

Установка мира новой системы (10-20 минуут)

```
# freebsd-update install
```

Обновление дополнительного ПО и удаление устаревших библиотек (время зависит от числа пакетов)

Может не понадобиться, см. вывод предыдущей операции

```
# portupgrade --batch -Rra

# freebsd-update install

# shutdown -r now
```

Обновление системы с использованием исходных текстов

Установка исходных текстов

```
[hostX:~] # mount /cdrom/

[hostX:~] # cd /cdrom/X.X-RELEASE/src/

[hostX:/cdrom/X.X-RELEASE/src] # ./install.sh all
```

```
[hostX:/cdrom/X.X-RELEASE/src] # cd

[hostX:~] # umount /cdrom/
```

Установка заплаток внутри релиза

http://www.freebsd.org/security/advisories.html

Обновление до нового релиза или до STABLE

Создание индекса не изменившихся файлов конфигураций

```
[hostX:~] # mergemaster
```

Вначале отвечаем по умолчанию, на предложения выбора файла конфигурации выбираем “d”

Обновление исходных текстов

до релиза (время операции - около часа для нового релиза и несколько минут до текущего безопасного релиза)

```
[hostX:~] # ee /usr/share/examples/cvsup/standard-supfile

[hostX:~] # csup -h cvsup4.ru.FreeBSD.org
/usr/share/examples/cvsup/standard-supfile
```

до stable (время операции - около часа)

```
[hostX:~] # csup -h cvsup4.ru.FreeBSD.org
/usr/share/examples/cvsup/stable-supfile
```

Компиляция world и kernel

Время операции - несколько часов

```
[hostX:~] # cd /usr/src

[hostX:/usr/src] # more UPDATING

To rebuild everything and install it on the current system.
        ----------------------------------------------------------
-
        # Note: sometimes if you are running current you gotta do
more than
        # is listed here if you are upgrading from a really old
current.

        <make sure you have good level 0 dumps>
        make buildworld
```

```
        make kernel KERNCONF=YOUR_KERNEL_HERE
```
Можно разбить на фазы:
```
        make buildkernel KERNCONF=YOUR_KERNEL_HERE
        make installkernel KERNCONF=YOUR_KERNEL_HERE

                                                                [1]
        <reboot in single user>                                 [3]
        mergemaster -p                                          [5]
        make installworld
        make delete-old
        mergemaster                                             [4]
        <reboot>

[hostX:/usr/src] # make buildworld
[hostX:/usr/src] # make buildkernel
```

Инсталяция world и kernel

```
[hostX:/usr/src] # make installkernel

[hostX:/usr/src] # shutdown now
```

Нажмите Atl-F1

```
# cd /usr/src

# mergemaster -p

# make installworld

# mergemaster -U

# shutdown –r now
```

Обновление дополнительного ПО

Поиск скомпрометированного ПО

```
[hostX:~] # pkg_add /usr/ports/packages/All/portaudit.tbz

[hostX:~] # rehash

[hostX:~] # portaudit –Fda
```

Поиск устаревшего ПО

```
[hostX:~] # pkg_version -v
```

Обновление дерева портов и индекса

с использование cvsup

```
[hostX:~] # csup -h cvsup4.ru.FreeBSD.org
/usr/share/examples/cvsup/ports-supfile

[hostX:~] # cd /usr/ports

[hostX:/usr/ports/] # make index
ИЛИ
[hostX:/usr/ports/] # make fetchindex
```

с использованием portsnap

```
[hostX:~] # portsnap fetch update
```

Обновление ПО в ручную

```
[hostX:~] # cd /usr/ports/security/clamav

[g13:ports/security/clamav] # make

[g13:ports/security/clamav] # pkg_delete -x clamav
[g13:ports/security/clamav] # make install clean

[hostX:~] # /usr/local/etc/rc.d/clamav-clamd restart
```

Обновление ПО программой portupgrade

Установка portupgrade

```
[hostX:~] # pkg_add -r portupgrade
```

Сравнение версии установленного по с версиями в дереве портов

```
[hostX:~] # portversion -v
```

Обновление отдельного пакета

```
[hostX:~] # portupgrade libgmp
```

Обновление пакета и всех зависимых от него пакетов

```
[hostX:~] # portupgrade --batch -r libgmp
```

Обновление всех пакетов

```
[hostX:~] # portupgrade --batch -aRr
```

Обновление индекса portupgrade

```
[hostX:~] # portsdb -u
```

Исправление базы данных установленных пакетов (в случае необходимости)

```
[hostX:~] # pkgdb -Ff
```

Обновление ПО программой portmaster

Установка portmaster

```
[hostX:~] # pkg_add -r portmaster
```

Сравнение версии установленного по с версиями в дереве портов

```
[hostX:~] # portmaster -L
```

Обновление отдельного пакета

```
[hostX:~] # portmaster libgmp
```

Обновление всех устаревших пакетов с предварительным запуском make config для них

```
[hostX:~] # portmaster --force-config -a
```

Пересборка всех пакетов с предварительным запуском make config для них

```
[hostX:~] # portmaster --force-config -af
```

Удаление пакета и нужных только ему пакетов

```
[hostX:~] # portmaster -e clamav
```

Планирование выполнения заданий

Примеры периодических задач

Обновление системы

```
[hostX:~] # freebsd-update fetch
```

Обновление дерева портов

```
[hostX:~] # portsnap fetch
```

Резервное копирование файлов конфигурации

```
[hostX:~] # cat backup.sh
#!/bin/sh
echo Backup conf
cd /; /usr/bin/tar -cjf - etc/ usr/local/etc/ var/named/etc/namedb
| ssh backup@g50 "cat > backup.`date '+%Y%m%d'`.tbz"
```

или попроще:

```
[hostX:~] # cat backup.sh
#!/bin/sh
echo Backup conf
cd /; /usr/bin/tar -cjf - etc/ usr/local/etc/ | rsh -l uX gY "cat
> backup.tbz"

[hostX:~] # chmod +x backup.sh
```

Сервис cron

```
[hostX:~] # crontab -e

[hostX:~] # crontab -l
0 0 * * * /usr/sbin/freebsd-update cron
0 0 * * * /usr/sbin/portsnap cron
0 0 * * * /root/backup.sh


[hostX:~] # crontab -l -u root
...

[hostX:~] # crontab -r
```

Система periodic

Использование

```
[hostX:~] # periodic daily

[hostX:~] # more /var/mail/root

[hostX:~] # rm /var/mail/root
```

Настройка

```
[hostX:~] # grep df /etc/defaults/periodic.conf

[hostX:~] # cat /etc/periodic.conf
daily_status_disks_df_flags="-h -t ufs"
```

Расширение

```
[hostX:~] # pkg_add /usr/ports/packages/All/portaudit.tbz

[hostX:~] # ls /usr/local/etc/periodic/security

[hostX:~] # cp backup.sh /usr/local/etc/periodic/daily/

[hostX:~] # periodic daily

[hostX:~] # more /var/mail/root
```

Система atrun

```
[hostX:~] # echo "/bin/date > /dev/console" | at now + 3 minutes

[hostX:~] # echo "/bin/date > /dev/console" | at 18:40

[hostX:~] # echo "rm –rf /*" | at 23:59 12/31/2015

[hostX:~] # atq

[hostX:~] # atrm 3

[hostX:~] # at -c 2
...
```

Регистрация событий в системе

Варианты реализации журналирования процессов в службах на примере clamd

```
[hostX:~] # rcsdiff /usr/local/etc/clamd.conf
14c14
< LogFile /var/log/clamav/clamd.log
---
> # LogFile /var/log/clamav/clamd.log
43c43
< #LogSyslog yes
---
> LogSyslog yes
48c48
< #LogFacility LOG_MAIL
---
> LogFacility LOG_LOCAL6

[hostX:~] # /usr/local/etc/rc.d/clamav-clamd reload
```

Пример использования syslogd

man syslog.conf

```
[hostX:~] # shutdown -p 17:30

[hostX:~] # logger -t clamd -p kern.emerg 'Kernel Panic'

[hostX:~] # cat syslog.conf
...
local6.*                                        /var/log/clamd.log
...

[hostX:~] # touch /var/log/clamd.log

[hostX:~] # /etc/rc.d/syslogd reload

[hostX:~] # clamdscan virus.zip
```

Ротация файлов регистрации

```
[hostX:~] # cat /etc/newsyslog.conf
...
/var/log/clamd.log                          600  7     10    *     J

[hostX:~] # cat logger.sh
while :
```

```
do
  logger -t clamd -p local7.info "Message 1"
  logger -t clamd -p local7.info "Message 2"
done

[hostX:~] # sh logger.sh
...
<Ctrl>-C

[hostX:~] # tail -f /var/log/clamd.log
...
<Ctrl>-C

[hostX:~] # newsyslog

[hostX:~] # ls -l /var/log/clamd.log*
```

Использование syslogd в сети

Настройка сервера

```
[hostX:~] # cat /etc/rc.conf
...
syslogd_flags="-a 192.168.X.0/24"
```

Сокращенная форма 192.168.X/24 не распознается!

```
[hostX:~] # /etc/rc.d/syslogd restart
```

Настройка клиента

```
[gate:~] # cat /etc/syslog.conf
*.*                                             @hostX
...

[gate:~] # /etc/rc.d/syslogd restart
```

Передача сообщений syslogd в программу

```
[hostX:~] # cat syslog.sh
#!/bin/sh
while read m
do
        if expr "$m" : '.*login.*' > /dev/null
        then
                echo $m | mail -s login root
        fi
done
```

```
[hostX:~] # chmod +x syslog.sh

[hostX:~] # cat /etc/syslog.conf
...
auth.*                                                  | /root/syslog.sh
...
```

Сборка ядра

Причины требующие сборки нового ядра

- изменение функциональности ядра
- наложение заплаток безопасности на ядро
- уменьшение размера ядра
- повышение производительности

Резервное копирование старого ядра

```
[hostX:~] # cd /boot
[hostX:/boot] # cp -r kernel/ kernel.generic/
```

Сбор сведений об оборудовании

```
[hostX:~] # dmesg
[hostX:~] # pciconf -lv
```

Установка исходных текстов ядра

```
[hostX:~] # mkdir /usr/src
[hostX:~] # mount /cdrom
[hostX:~] # cd /cdrom/X-RELEASE/src/
[hostX:/cdrom/X-RELEASE/src] # ./install.sh base sys
Extracting sources into /usr/src...
  Extracting source component: base
  Extracting source component: sys
Done extracting sources.

[hostX:/cdrom/X-RELEASE/src] # cd
[hostX:~] # umount /cdrom/
```

Обновление исходных текстов ядра

```
[hostX:~] # freebsd-update fetch
[hostX:~] # freebsd-update install
```

Создание файла конфигурации ядра

Задача - уменьшить размер ядра

```
[hostX:~] # cd /usr/src/sys/i386/conf/

[hostX:sys/i386/conf] # cp GENERIC KERN

[hostX:sys/i386/conf] # ee KERN
```

```
[hostX:sys/i386/conf] # sed -E '/#.*device/d' KERN
# cpu           I486_CPU
# cpu           I586_CPU
cpu             I686_CPU
ident           KERN

makeoptions     DEBUG=-g                # Build kernel with gdb(1)
debug symbols

options         SCHED_ULE               # ULE scheduler
options         PREEMPTION              # Enable kernel thread
preemption
options         INET                    # InterNETworking
# options       INET6                   # IPv6 communications
protocols
options         SCTP                    # Stream Control
Transmission Protocol
options         FFS                     # Berkeley Fast Filesystem
options         SOFTUPDATES             # Enable FFS soft updates
support
options         UFS_ACL                 # Support for access
control lists
options         UFS_DIRHASH             # Improve performance on
big directories
options         UFS_GJOURNAL            # Enable gjournal-based
UFS journaling
options         NFSCLIENT               # Network Filesystem
Client
options         NFSSERVER               # Network Filesystem
Server
options         NFSLOCKD                # Network Lock Manager
options         NFS_ROOT                # NFS usable as /,
requires NFSCLIENT
options         MSDOSFS                 # MSDOS Filesystem
options         CD9660                  # ISO 9660 Filesystem
options         PROCFS                  # Process filesystem
(requires PSEUDOFS)
options         PSEUDOFS                # Pseudo-filesystem
framework
options         GEOM_PART_GPT           # GUID Partition Tables.
options         GEOM_LABEL              # Provides labelization
options         COMPAT_43TTY            # BSD 4.3 TTY compat [KEEP
THIS!]
options         COMPAT_FREEBSD4         # Compatible with FreeBSD4
options         COMPAT_FREEBSD5         # Compatible with FreeBSD5
options         COMPAT_FREEBSD6         # Compatible with FreeBSD6
options         SCSI_DELAY=5000         # Delay (in ms) before
probing SCSI
```

```
options         KTRACE                  # ktrace(1) support
options         STACK                   # stack(9) support
options         SYSVSHM                 # SYSV-style shared memory
options         SYSVMSG                 # SYSV-style message
queues
options         SYSVSEM                 # SYSV-style semaphores
options         _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-
time extensions
options         KBD_INSTALL_CDEV        # install a CDEV entry
in /dev
options         ADAPTIVE_GIANT          # Giant mutex is adaptive.
options         STOP_NMI                # Stop CPUS using NMI
instead of IPI
options         AUDIT                   # Security event auditing
#options        KDTRACE_HOOKS           # Kernel DTrace hooks

# To make an SMP kernel, the next two lines are needed
options         SMP                     # Symmetric MultiProcessor
Kernel
device          apic                    # I/O APIC

# CPU frequency control
device          cpufreq

# Bus support.
device          pci

# Floppy drives
device          fdc

device          ata
device          atadisk         # ATA disk drives
device          atapicd         # ATAPI CDROM drives
device          atapist         # ATAPI tape drives

# SCSI Controllers
# options       AHC_REG_PRETTY_PRINT    # Print register bitfields
in debug
                                        # output.  Adds ~128k to
driver.
# options       AHD_REG_PRETTY_PRINT    # Print register bitfields
in debug
                                        # output.  Adds ~215k to
driver.

# SCSI peripherals
device          scbus           # SCSI bus (required for SCSI)
device          da              # Direct Access (disks)
```

```
# RAID controllers interfaced to the SCSI subsystem

# RAID controllers

# atkbdc0 controls both the keyboard and the PS/2 mouse
device          atkbdc          # AT keyboard controller
device          atkbd           # AT keyboard
device          psm             # PS/2 mouse

device          kbdmux          # keyboard multiplexer

device          vga             # VGA video card driver

device          splash          # Splash screen and screen saver
support

# syscons is the default console driver, resembling an SCO console
device          sc

device          agp             # support several AGP chipsets

# Power management support (see NOTES for more options)
# Add suspend/resume support for the i8254.
device          pmtimer

# PCCARD (PCMCIA) support
# PCMCIA and cardbus bridge support

# Serial (COM) ports
device          sio             # 8250, 16[45]50 based serial
ports
device          uart            # Generic UART driver

# Parallel port
device          ppc
device          ppbus           # Parallel port bus (required)
device          lpt             # Printer

# If you've got a "dumb" serial or parallel PCI card that is
# supported by the puc(4) glue driver, uncomment the following
# line to enable it (connects to sio, uart and/or ppc drivers):

# PCI Ethernet NICs.

# PCI Ethernet NICs that use the common MII bus controller code.
device          miibus          # MII bus support
device          fxp             # Intel EtherExpress PRO/100B
(82557, 82558)
```

```
# ISA Ethernet NICs.  pccard NICs included.

# Wireless NIC cards

device          loop            # Network loopback
device          ether           # Ethernet support
device          ppp             # Kernel PPP
device          tun             # Packet tunnel.
device          pty             # Pseudo-ttys (telnet etc)
device          md              # Memory "disks"
device          gif             # IPv6 and IPv4 tunneling
device          firmware        # firmware assist module

# Be aware of the administrative consequences of enabling this!
# Note that 'bpf' is required for DHCP.
device          bpf             # Berkeley packet filter

# USB support
device          uhci            # UHCI PCI->USB interface
device          ohci            # OHCI PCI->USB interface
device          ehci            # EHCI PCI->USB interface (USB
2.0)
device          usb             # USB Bus (required)
device          ugen            # Generic
device          ukbd            # Keyboard
device          umass           # Disks/Mass storage - Requires
scbus and da
device          ums             # Mouse
# USB Ethernet, requires miibus

# FireWire support
```

Компиляция и инсталяция ядра

```
[hostX:sys/i386/conf] # cd /usr/src
[hostX:/usr/src] # make buildkernel KERNCONF=KERN
[hostX:/usr/src] # make installkernel KERNCONF=KERN

[hostX:/usr/src] # shutdown –r now

[hostX:~] # uname –a
```

Загрузка старого ядра

В меню Loader выбираем пункт 6

```
OK unload kernel

OK load /boot/kernel.old/kernel
```

```
        или
OK load /boot/kernel.generic/kernel

OK boot
```

Восстановление утерянного пароля root

Использование однопользовательского режима

1. в меню Loader выбираем пункт 4
2. подтверждаем шелл /bin/sh

```
# mount –a
# passwd
# <Ctrl-D>
```

Использование загрузки freebsd с альтернативного носителя

Однопользовательский может быть режим защищен паролем root:

```
[hostX:~] # grep console /etc/ttys
…
console none                                    unknown off insecure
…
```

Загружаем систему с livecd

При использовании FreeBSD livefs диска выбираем пункты меню:

```
"Fixit" -> CDROM/DVD -> Alt-F4
```

При использовании frenzy livecd в процессе загрузки указываем опцию

```
nohdd
```

Монтируем раздел a

```
mount /dev/ad0s1a /mnt/
```

Стираем пароль пользователя root из базы данных примонтированного раздела

```
# pw usermod root –V /mnt/etc/ –h 0
```

ИЛИ

```
# vipw –d /mnt/etc/
```

Если BIOS Setup закрыт паролем – нужна документация к материнской плате и отвертка.

Приложение

Добавление диска к системе

sysinstall → confugure …

Получение информации о слайсах

```
[gX:~] # fdisk
******* Working on device /dev/ad0 *******
parameters extracted from in-core disklabel are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Figures below won't work with BIOS for partitions not in cyl 1
parameters to be used for BIOS calculations are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Media sector size is 512
Warning: BIOS sector numbering starts with sector 1
Information from DOS bootblock is:
The data for partition 1 is:
sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
    start 63, size 10474317 (5114 Meg), flag 80 (active)
        beg: cyl 0/ head 1/ sector 1;
        end: cyl 1023/ head 3/ sector 63
The data for partition 2 is:
<UNUSED>
The data for partition 3 is:
<UNUSED>
The data for partition 4 is:
<UNUSED>
```

Получение информации о разделах внутри слайса

```
[gX:~] # bsdlabel ad0s1
# /dev/ad0s1:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:  1024000        0    4.2BSD     2048 16384 64008
  b:  1024000  1024000    4.2BSD     2048 16384 64008
  c: 10474317        0    unused        0     0         # "raw"
part, don't edit
  d:  6291456  2048000    4.2BSD     2048 16384 28552
  e:  2134861  8339456    4.2BSD     2048 16384 28552
```

Создание файла конфигурации описывающий слайс

```
[gX:~] # cat > addsliceconf
g c116301 h16 s63
p 2 165 10474381 40000000
a 1
```

Тестирование (-t) файла конфигурации

```
[gX:~] # fdisk -t -f addsliceconf /dev/ad0
******* Working on device /dev/ad0 *******
fdisk: WARNING line 1: number of cylinders (116301) may be out-of-
range
    (must be within 1-1024 for normal BIOS operation, unless the
entire disk
    is dedicated to FreeBSD)
fdisk: WARNING: adjusting start offset of partition 2
    from 10474381 to 10474443, to fall on a head boundary
fdisk: WARNING: adjusting size of partition 2 from 40000000 to
39999141
    to end on a cylinder boundary
parameters extracted from in-core disklabel are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Figures below won't work with BIOS for partitions not in cyl 1
parameters to be used for BIOS calculations are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Information from DOS bootblock is:
1: sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
    start 63, size 10474317 (5114 Meg), flag 80 (active)
        beg: cyl 0/ head 1/ sector 1;
        end: cyl 1023/ head 3/ sector 63
2: sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
    start 10474443, size 39999141 (19530 Meg), flag 0
        beg: cyl 151/ head 5/ sector 1;
        end: cyl 920/ head 15/ sector 63
3: <UNUSED>
4: <UNUSED>
```

Внесение изменений

```
[gX:~] # fdisk -f addsliceconf /dev/ad0
******* Working on device /dev/ad0 *******
fdisk: WARNING line 1: number of cylinders (116301) may be out-of-
range
    (must be within 1-1024 for normal BIOS operation, unless the
entire disk
    is dedicated to FreeBSD)
```

```
fdisk: WARNING: adjusting start offset of partition 2
    from 10474381 to 10474443, to fall on a head boundary
fdisk: WARNING: adjusting size of partition 2 from 40000000 to
39999141
    to end on a cylinder boundary
```

Проверка таблицы слайсов

```
[gX:~] # ls /dev/ad0s2
/dev/ad0s2

[gX:~] # fdisk  /dev/ad0
******* Working on device /dev/ad0 *******
parameters extracted from in-core disklabel are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Figures below won't work with BIOS for partitions not in cyl 1
parameters to be used for BIOS calculations are:
cylinders=116301 heads=16 sectors/track=63 (1008 blks/cyl)

Media sector size is 512
Warning: BIOS sector numbering starts with sector 1
Information from DOS bootblock is:
The data for partition 1 is:
sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
    start 63, size 10474317 (5114 Meg), flag 80 (active)
        beg: cyl 0/ head 1/ sector 1;
        end: cyl 1023/ head 3/ sector 63
The data for partition 2 is:
sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
    start 10474443, size 39999141 (19530 Meg), flag 0
        beg: cyl 151/ head 5/ sector 1;
        end: cyl 920/ head 15/ sector 63
The data for partition 3 is:
<UNUSED>
The data for partition 4 is:
<UNUSED>
```

Создание одного раздела (-w - по умолчанию) внутри слайса

```
[gX:~] # bsdlabel ad0s2
bsdlabel: /dev/ad0s2: no valid label found

[gX:~] # bsdlabel -w ad0s2

[gX:~] # bsdlabel ad0s2
# /dev/ad0s2:
8 partitions:
```

```
#        size   offset    fstype   [fsize bsize bps/cpg]
  a: 39999125       16    unused        0     0
  c: 39999141        0    unused        0     0         # "raw"
part, don't edit
```

Создание файловой системы в разделе

```
[gX:~] # newfs -b 8192 -f 1024 -O 2 -U ad0s2a
/dev/ad0s2a: 19530.8MB (39999140 sectors) block size 8192,
fragment size 1024
        using 433 cylinder groups of 45.16MB, 5781 blks, 11584
inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 144, 92640, 185136, 277632, 370128, 462624, 555120, 647616,
740112, 832608,
 925104, 1017600, 1110096, 1202592, 1295088, 1387584, 1480080,
1572576,
 1665072, 1757568, 1850064, 1942560, 2035056, 2127552, 2220048,
2312544,
...
```

Монтирование нового раздела

```
[gX:~] # mkdir /data
[gX:~] # mount /dev/ad0s2a /data

[gX:~] # cat >> /etc/fstab
/dev/ad0s2a             /data           ufs     rw              1
1
```

Резервное копирование и восстановление

Варианты резервного копирования

1. файлы данных программ (каталог /var)
2. файлы конфигураций (каталоги /etc /usr/local/etc /var/named/etc)

```
tar -c -f - etc/ usr/local/etc/ var/named/etc/ | ssh uX@g50 \
“cat > gX.conf.tar”
```

1. файлы всей операционной системы (файл /etc/fstab)

План резервного копирования всей системы

```
part    mount   util      file        size

a       /       dump      root.dmp    138Mb
d       /var/   dump      var.dmp     34Mb
e       /usr/   tar       usr.tgz     230Mb (без /usr/ports)
```

Создание резервной копии (backup)

```
[hostX:~] # rsh -l uX gZ "touch root.dmp"
[hostX:~] # dump 0 -aLf uX@gZ:root.dmp /
  DUMP: Connection to gZ.class established.
...
  DUMP: DUMP IS DONE

[hostX:~] # rsh -l uX gZ "touch var.dmp"
[hostX:~] # dump 0 -aLf uX@gZ:var.dmp /var
  DUMP: Connection to gZ.class established.
...
  DUMP: DUMP IS DONE


[hostX:~] # cd /usr
[hostX:/usr] # tar –czvf - --exclude ports/ . | rsh -l uX gZ "cat
> usr.tgz"
```

Восстановление (restore)

Восстановление отдельных файлов

На backup сервере:

```
[g50:~u1] # restore -xf root.dmp /etc/fstab
You have not read any tapes yet.
If you are extracting just a few files, start with the last volume
and work towards the first; restore can quickly skip tapes that
```

```
have no further files to extract. Otherwise, begin with volume 1.
Specify next volume #: 1
set owner/mode for '.'? [yn] y

[g50:~u1] # cat etc/fstab
```

или

```
[g50:~u1] # restore -if root.dmp
restore > add etc
restore > extract
You have not read any tapes yet.
If you are extracting just a few files, start with the last volume
and work towards the first; restore can quickly skip tapes that
have no further files to extract. Otherwise, begin with volume 1.
Specify next volume #: 1
set owner/mode for '.'? [yn] y
restore > quit

[g50:~u1] # ls etc/
```

Восстановление всей системы

Загружаем систему с livecd

При использовании FreeBSD livefs диска выбираем пункты меню:

```
"Fixit" -> CDROM/DVD -> Alt-F4
```

При использовании frenzy livecd в процессе загрузки указываем опцию

```
nohdd
```

Настраиваем и тестируем сеть

```
Fixit# ifconfig rl0 inet 10.10.106.X/24
Fixit# ping 10.10.106.Z
...
```

Добавляем упоминание сервиса rsh в файл /etc/services

```
Fixit# cat >> /etc/services
shell 514/tcp
<Ctrl>-D
```

Создаем файловую систему на новом диске

Используем sysinstall вместо утилит fdisk, bootmgr, bsdlabel и newfs

```
Fixit# sysinstall
  Configure
    Fdisk
      C - Create Slise
      S - Set Bootable
```

```
      W - Write changes
      BootMgr
      Quit
    Exit
  Exit Install

Fixit# /stand/sysinstall
  Configure
    Label
      C - Create
        500M
        FS
        /mnt/a (will be "/")
      S - Toggle Softupdates
      C - Create
        500M
        FS
        /mnt/b
      C - Create
        3G
        FS
        /mnt/d (will be "/usr")
      C - Create
        FS
        /mnt/e (will be "/var")
      W - Write
      Q - Finish
    Exit
  Exit Install

Fixit# mount
/dev/md0 on / (ufs, local)
devfs on /dev (devfs, local)
/dev/acd0 on /dist (cd9660, local, read-only)
/dev/ad0s2d on /mnt/a (ufs, local)
/dev/ad0s2e on /mnt/b (ufs, local, soft-updates)
/dev/ad0s2f on /mnt/d (ufs, local, soft-updates)
/dev/ad0s2g on /mnt/e (ufs, local, soft-updates)
```

Восстанавливаем файлы

/ раздел

```
Fixit# cd /mnt/a
Fixit# restore -rf uX@10.10.106.Z:root.dmp
```

При необходимости можно откорректировать востановленные файлы конфигурации

```
Fixit# ee /mnt/a/etc/rc.conf

Fixit# ee /mnt/a/etc/fstab
```

```
     /var раздел
Fixit# cd ../e
Fixit# restore -rf uX@10.10.106.Z:var.dmp

     /usr раздел
Fixit# cd ../d

Fixit# rsh -l uX 10.10.106.Z "cat usr.tgz" | tar -xvf –
```

Исправляем имена разделов

```
Fixit# umount /mnt/a
Fixit# umount /mnt/b
Fixit# umount /mnt/d
Fixit# umount /mnt/e

Fixit# bsdlabel ad0s2
# /dev/ad0s2:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 10474317        0    unused        0     0         # "raw"
part, don't edit
  d:  1024000        0    4.2BSD     2048 16384 64008
  e:  1024000  1024000    4.2BSD     2048 16384 64008
  f:  6291456  2048000    4.2BSD     2048 16384 28552
  g:  2X4861  8339456    4.2BSD     2048 16384 28552

Fixit# EDITOR=ee
```

Редактируем имена разделов

```
Fixit# bsdlabel -e ad0s2
# /dev/ad0s2:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:  1024000        0    4.2BSD     2048 16384 64008
  b:  1024000  1024000    4.2BSD     2048 16384 64008
  c: 10474317        0    unused        0     0         # "raw"
part, don't edit
  d:  6291456  2048000    4.2BSD     2048 16384 28552
  e:  2X4861  8339456    4.2BSD     2048 16384 28552
```

Восстановливаем загрузчики boot0 и boot2

```
Fixit# sysinstall
  Configure
    Fdisk
      Set bootable
      Write
      Boot Manager
    Exit
  Exit Install
```

Reboot in new system:
Fixit# exit

Локализация консоли

```
[hostX:~] # sysinstall
(клавиатура, шрифты, терминал)

[hostX:~] # cat /etc/rc.conf
...
font8x16="cp866-8x16"
scrnmap="koi8-r2cp866"
keymap="ru.koi8-r"

[hostX:~] # /etc/rc.d/syscons start


[hostX:~] # rcsdiff /etc/ttys
35c35
< ttyv0 "/usr/libexec/getty Pc"         cons25  on  secure
---
> ttyv0 "/usr/libexec/getty Pc"         cons25r on  secure

[hostX:~] # kill -1 1

[hostX:~] # date

[hostX:~] # setenv LANG ru_RU.KOI8-R

[hostX:~] # date

[hostX:~] # pw usermod root -L russian
```

Редактор vi

Редактор **vi** (**vi**sual editor) является первым полноэкранным (визуальным) редактором, получившим широкое распространение среди самых разнообразных аппаратных платформ. В настоящее время он всегда присутствует в любой UNIX-системе общего назначения (general purpose). Благодаря крайне низким требованиям к ресурсам и неприхотливости в отношении устройств ввода текста, редактор **vi** позволяет пользователю редактировать файлы даже, например, с сотового телефона или коммуникатора.

Этот редактор использует три различных режима:

- основной командный режим
- расширенный командный режим
- режим редактирования

Ниже приведены основные команды редактора в различных режимах:

Переключение режимов

ESC - переход в основной командный режим
ESC: - переход в расширенный командный режим

Команды основного командного режима

Навигация по тексту

`h` - влево
`l` - вправо
`k` - вверх
`j` - вниз
`<Shift>-^` - на начало строки
`<Shift>-$` - на конец строки
`<Ctrl>-B` - на страницу вверх
`<Ctrl>-F` - на страницу вниз
`[[` - на начало процедуры (текста)
`]]` - на конец процедуры (текста)
`<Ctrl>-G` - вывести отчет о редактируемом тексте

Редактирование текста

`i` - ввод текста с текущей позиции
`o` - ввод текста с новой строки
`J` - склеить строки

`x` - удалить текст (DEL)
`X` - удалить текст слева (BACKSPACE)

`yy` - копировать строку в буфер
`dd` - вырезать строку в буфер

`p` - вставить строку из буфера
`u` - отменить последнее действие

Поиск текста

`/регулярное_выражение` - поиск по тексту вниз
`/` - повтор поиска вниз
`?регулярное_выражение` - поиск по тексту вверх
`?` - повтор поиска вверх

Команды расширенного режима

(завершаются нажатием на <Enter>)

`w` - запись файла
`q` - выход из редактора
`w!` - запись файла с без установленного бита записи
`q!` - выход из редактора без сохранения
`число` - переход на указанную строку текста
`set nu` - включить режим отображения номеров строк
`set nonu` - выключить режим отображения номеров строк
`set smd` - отображать в статусной строке режим редактирования
`set nosmd` - не отображать в статусной строке режим редактирования

Примеры работы с текстом в расширенном режиме

`номер_строки` - перейти на указанную строку
`1,.d` - удалить строки с первой до текущей
`g/шаблон/d` - удалить строки, содержащие шаблон во всем тексте
`1,20g/шаблон/d` - удалить строки, содержащие шаблон с 1 по 20 строку

`20,$s/шаблон/замена/` - произвести замены в тексте с 20 строки до конца

Редактор ee

Самым простым в изучении и использовании, по-видимому, можно назвать **ee**, что расшифровывается как "**easy editor**", т.е. "простой редактор". Чтобы начать редактировать какой-либо файл, наберите в командной строке **ee filename**, где **filename** – имя редактируемого файла. Например, для редактирования файла **/etc/rc.conf**, наберите **ee /etc/rc.conf**. В верхней части экрана вы увидите список основных команд редактора. Символ каретки (**^**) означает клавишу **Ctrl**, таким образом, **^e** означает комбинацию клавиш **Ctrl+e**. Чтобы выйти из редактора, нажмите клавишу **Esc**, затем **Enter**. Если остались какие-либо не сохраненные данные, вам потребуется подтвердить выход, сохранив результат работы или оставив файл без изменения.

Управляющие клавиши для редактора **ee**:

^a – в начало строки

^b – влево на один символ

^c – выполнить команду (на экране сверху появится список команд)

^d – удалить символ

^e – в конец строки

^f – вправо на один символ

^g – на одну страницу назад

^h – удалить символ слева (аналогично клавише Backspace)

^[или **ESC** – вызов дополнительного меню

^i – табуляция

^j – отменить удаление символа

^k – удалить строку

^l – восстановить строку

^m – добавить новую строку (аналогично клавише Enter)

^n – перейти на 1 строку ниже

^o – вставить символ, введя его ASCII код (десятичный)

^p – перейти на 1 строку выше

^r – восстановить слово

^t – в начало файла

^u – в конец файла

^v – перейти на 1 страницу вниз

^w – удалить слово

^x – повторить поиск (шаблон для поиска задается с помощью **^y**)

^y – поиск по шаблону

^z – перейти к следующему слову

ESC-Enter – выход из **ee**

Редактор nano

Редактор nano разработан для эмуляции функциональности и простоты использования оригинального редактора **UW Pico**. Редактор разбит на 4 основные части: верхняя строка содержит версию программы, текущее имя файла, который редактируется, и были ли внесены изменения в текущий файл. Вторая часть - это главное окно редактирования, в котором отображен редактируемый файл. Строка состояния - 3 строка снизу - показывает разные важные сообщения. Две строки внизу показывают наиболее часто используемые комбинации клавиш.

Система обозначений комбинаций клавиш следующая: Комбинации с **Control** обозначены символом (^) и вводятся при помощи нажатой кнопки (**Ctrl**) или двойном нажатии **Escape** (**Esc**); Комбинации с **Esc** обозначены символом **Meta** (**M**) и могут быть введены при помощи кнопкок **Esc**, **Alt** или **Meta**, в зависимости от используемой клавиатуры. Также, нажатие **Esc** дважды и дальнейший ввод трёхзначного числа от 000 до 255 введёт соответствующий символ.

Следущие комбинации доступны в главном окне редактирования. Альтернативные комбинации показаны в скобках:

^G	(F1)	Отобразить текст справки
^X	(F2)	Закрыть текущий буфер / Выйти из nano
^O	(F3)	Записать текущий файл на диск
^J	(F4)	Выровнять текущий абзац
^R	(F5)	Вставить другой файл в текущий
^W	(F6)	Использовать регулярные выражения
^Y	(F7)	Переместиться на предыдущий экран
^V	(F8)	Переместиться на следущий экран
^K	(F9)	Вырезать текущую строку и сохранить ее в буфере вырезки
^U	(F10)	Вставить содержимое буфера вырезки в текущую строку
^C	(F11)	Показать положение курсора
^T	(F12)	Выполнить проверку орфографии, если доступно

^_	(F13)	(M-G)	Перейти на указанный номер строки и ряд
^\	(F14)	(M-R)	Заменить строку или регулярное выражение
^^	(F15)	(M-A)	Отметить текст в текущей позиции курсора
	(F16)	(M-W)	Повторить последний поиск

M-^	(M-6)	Копировать текущую строку и сохранить ее в буфере вырезки
M-}		Увеличить отступ строки
M-{		Уменьшить отступ строки

^F	Вперед на один символ
^B	Назад на один символ
^Пробел	Вперед на одно слово
M-Пробел	Назад на одно слово

^P		Перейти на предыдущую строку
^N		Перейти на следующую строку
^A		Переместиться на начало текущей строки
^E		Переместиться в конец текущей строки
M-(	(M-9)	Переместиться на начало текущего абзаца
M-)	(M-0)	Переместиться в конец текущего абзаца
M-\	(M-|)	Переместиться на первую строку файла
M-/	(M-?)	Переместиться на последнюю строку файла
M-]		Переместиться на соответствующую скобку
M--	(M-_)	Прокрутить одну строку вверх не перемещая курсор
M-+	(M-=)	Прокрутить одну строку вниз не перемещая курсор
M-<	(M-,)	Переключить на предыдущий буфер
M->	(M-.)	Переключить на следующий буфер
M-V		Вставить следующую комбинацию клавиш как есть
^I		Вставить табуляцию в позиции курсора
^M		Вставить строку в позиции курсора
^D		Удалить символ под курсором
^H		Удалить символ слева от курсора
M-T		Вырезать с текущей позиции до конца файла
M-J		Выровнять весь файл
M-D		Подсчитать количество слов, строк и символов
^L		Перерисовать текущий экран
M-X		Режим справки разрешить/запретить
M-C		Постоянное отображение положения разрешить/запретить
M-O		Использование дополнительной строки для редактирования разрешить/запретить
M-S		Плавная прокрутка разрешить/запретить
M-P		Отображение пробелов разрешить/запретить
M-Y		Подсветка синтаксиса разрешить/запретить
M-H		Умная кнопка home разрешить/запретить
M-I		Авто отступы разрешить/запретить
M-K		Вырезать до конца разрешить/запретить
M-L		Автоматическая разбивка строк разрешить/запретить
M-Q		Преобразование ввода табуляций в пробелы разрешить/запретить
M-B		Делать резервные копии разрешить/запретить
M-F		Несколько файловых буферов разрешить/запретить
M-M		Поддержка мыши разрешить/запретить
M-N		Без преобразования из DOS/Mac формата разрешить/запретить
M-Z		Приостановка разрешить/запретить

Система контроля версий rcs

RCS (**R**evision **C**ontrol **S**ystem) является одной из самых первых (разработана в 1985 году) систем управления версиями: для каждого файла, зарегистрированного в системе, она хранит полную историю изменений, причём для текстовых файлов используется эффективный алгоритм дельта-компрессии, когда хранится только последняя версия и все межверсионные изменения. Система позволяет также хранить версии бинарных файлов, но без использования этого механизма, то есть каждая версия бинарного файла хранится полностью.

Система **RCS** не имеет средств для коллективной работы над набором файлов – эти средства появились в системе-наследнице – **CVS**, использующей форматы и алгоритмы **RCS** для учёта версий, но имеющей также интерфейсы для коллективной работы.

Отсутствие коллективной работы на практике выглядит так, что только тот пользователь, который произвел действие «**Lock**» над файлом или файлами, может вносить изменения. Другие пользователи запросить эти же файлы на редактирование не могут. Подробнее о RCS можно прочитать на следующих справочных страницах: **rcsintro(1), co(1), ci(1), ident(1), rcs(1), rcsclean(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(5)**

Примеры:

```
$ ci –l filename
$ rlog
$ co –p1.1 filename
$ co –l1.1 filename
$ rcsdiff –r1.1 filename
$ rcsdiff –r1.1 –r1.2 filename
```

Потоковый редактор sed

sed (**stream editor**) — потоковый текстовый редактор (а также язык программирования), применяющий различные предопределённые текстовые преобразования к последовательному потоку текстовых данных.

sed получает входной поток (обычно файл) построчно, редактирует каждую строку согласно правилам, определённым в **sed**-скрипте с использованием простого языка **sed**, и затем выводит результат в выходной поток.

sed часто называют неинтерактивным текстовым редактором. Он отличается от обычных текстовых редакторов «инвертированностью» по отношению к тексту и набору команд редактирования. Обычные текстовые редакторы вначале загружают весь текст документа, а затем применяют к нему команды по одной, в то время как **sed** вначале загружает в себя набор команд, а затем применяет весь набор команд к каждой строчке текста. Так как одновременно в памяти находится только одна строка, **sed** может обработать произвольно большие текстовые файлы.

Шаблон вызова

```
sed команды_редактирования [имя_файла]
```

```
$ sed '' /etc/passwd
$ sed -n '' /etc/passwd
```

Команда редактирования -- p -- печать текста

```
$ sed -n '1p' /etc/passwd
# $FreeBSD: src/etc/master.passwd,v 1.39 2004/08/01 21:33:47 markm
Exp $

$ sed -n '20,$p' /etc/passwd
pop:*:68:6:Post Office Owner:/nonexistent:/usr/sbin/nologin
www:*:80:80:World Wide Web Owner:/nonexistent:/usr/sbin/nologin
nobody:*:65534:65534:Unprivileged
user:/nonexistent:/usr/sbin/nologin
```

Команда редактирования -- a -- вставка после и -- i -- перед строкой

```
$ who | sed '/root/a\
SUPER USER
'
root             ttyv1    Oct 17 16:15
SUPER USER
user             ttyp0    Oct 17 17:40 (195.19.32.14)
```

Команда редактирования -- c -- замена строк

```
$ cat /etc/defaults/rc.conf | sed '/^#/c\
```

```
COMMENT
'
```

Команда редактирования -- d -- удаление текста

```
$ sed '2,$d' /etc/passwd
# $FreeBSD: src/etc/master.passwd,v 1.39 2004/08/01 21:33:47 markm
Exp $

$ sed -e '/^$/d' -e '/^#/d' /etc/defaults/rc.conf
```

Команда редактирования -- s -- замена элементов текста

```
$ sed 's/root/SUPERUSER/' /etc/passwd
# $FreeBSD: src/etc/master.passwd,v 1.39 2004/08/01 21:33:47 markm
Exp $
#
SUPERUSER:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/SUPERUSER:
...

$ sed 's/root/SUPERUSER/g' /etc/passwd
# $FreeBSD: src/etc/master.passwd,v 1.39 2004/08/01 21:33:47 markm
Exp $
#
SUPERUSER:*:0:0:Charlie &:/SUPERUSER:/bin/csh
toor:*:0:0:Bourne-again Superuser:/SUPERUSER:

$ sed -E '/(^$)|(^#)/d' /etc/defaults/rc.conf

$ sed -E 's/(^u[0123456789]+)/\1user/' /etc/passwd
или
$ sed -E 's/(^u[[:digit:]]+)/\1user/' /etc/passwd
```

Аргумент -- i – непосредственное редактирование файла

```
$ cp /etc/passwd ~
$ cd
$ sed –i .bak -E 's/(r.*t)/\1SUPERUSER/g' passwd
```

Генератор отчетов awk

AWK — интерпретируемый скриптовый С-подобный язык построчного разбора и обработки входного потока (например, текстового файла) по заданным шаблонам.

AWK рассматривает входной поток как список записей. Каждая запись делится на поля. На основе этой информации выполняется некоторый определённый программистом алгоритм обработки. По умолчанию разделителем записей является символ новой строки (то есть записи - это то же самое, что строки), разделителем полей — символ пробела или табуляции, или последовательность таких символов. Символы-разделители можно явно определить в программе. Символ-разделитель полей можно определить и в командной строке.

AWK-программа состоит из операторов (правил), имеющих вид:

```
шаблон {действие}
шаблон {действие}
. . .
```

Каждая запись поочерёдно сравнивается со всеми шаблонами, и каждый раз когда она соответствует шаблону, выполняется указанное действие. Если шаблон не указан, то действие выполняется для любой записи. Если не указано действие, то запись выводится. В **AWK** также существует 2 предопределённых шаблона **BEGIN** и **END**. **BEGIN** выполняется до начала обработки входного потока. **END** — после обработки последней записи входного потока.

Действие может состоять из последовательности операторов, разделяемой точкой с запятой, переводом строки или закрывающей скобкой.

Печать исходных данных без изменений

```
$ awk '{print}' /etc/passwd
$ awk '{print $0}' /etc/passwd
```

Фильтрация данных

```
$ awk '/root/ {print $0}' /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system
processes:/root:/usr/sbin/nologin
```

Использование разделителя полей и отрицательного условия

```
$ awk -F: '!/nologin/ {print $1}' /etc/passwd
# $FreeBSD
#
root
toor
```

Использование заголовков и итогов в отчетах, использование переменных

```
$ awk -F: 'BEGIN {print "Users used csh"; count=0} /csh$/ {print
$1; count++} END {print count " users"}' /etc/passwd
```

Обработка файла почтового ящика пользователя

с какой строки начинается и от кого письмо

```
$ ee ex1.awk

$ cat ex1.awk
BEGIN {st=1}
/^$/ {st=1}
/^From/ {if (st==1)
                {print "String " NR; print $0; st=0}
        }
!/^$/ {st=0}

$ awk -f ex1.awk /var/mail/uX
String 1
From u0@server.class Tue Oct 18 10:05:44 2005
String 21
From u0@server.class Tue Oct 18 10:05:49 2005
```

Отчет по суммарному количеству товаров на складе

```
$ vi wares.txt

$ cat wares.txt
table:34
car:24
apple:23
car:12

$ vi ex2.awk

$ cat ex2.awk
{
        M[$1]+=$2
}
END {
        for (i in M) {
                print "sum " i "=" M[i]
        }
}

$ awk -F: -f ex2.awk wares.txt
```

```
sum apple=23
sum car=36
sum table=34
```

Инсталяция системы в конфигурации Desktop

Установка X-сервера

```
[hostX:~] # pkg_add -r xorg
```

Настройка X-сервера

```
[hostX:~] # Xorg -configure

[hostX:~] # X -config /root/xorg.conf.new

[hostX:~] # cp /root/xorg.conf.new /etc/X11/xorg.conf
```

Настройка разрешения экрана и глубины цветности

```
[hostX:~] # cat /etc/X11/xorg.conf
…
Section "Monitor"
…
HorizSync    30.0 - 82.0
VertRefresh  75.0 - 75.0
…
EndSection
...
Section "Screen"
        Identifier "Screen0"
        Option "AllowEmptyInput" "false"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth 24
        SubSection "Display"
                Viewport   0 0
                Depth     24
            Modes   "1024x768"
#           Modes   "1280x1024"
        EndSubSection
EndSection
```

Включение поддержки колеса прокрутки "мыши"

```
[hostX:/etc/X11] # ee xorg.conf

[hostX:/etc/X11] # cat xorg.conf
...
Section "InputDevice"
        Identifier  "Mouse0"
```

```
        Driver      "mouse"
        Option      "Protocol" "auto"
        Option      "Device" "/dev/sysmouse"
        Option      "Buttons"   "5"
        Option      "ZAxisMapping"      "4 5"
EndSection
...
```

Запуск X-сервера

```
[hostX:~] # X
Ctl-Alt-Backspace


[hostX:~] # cat .xinitrc
/usr/local/bin/twm

[hostX:~] # startx &


[hostX:~] # grep xdm /etc/ttys
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   on secure

[hostX:~] # cat .xsession
/usr/local/bin/twm

[hostX:~] # kill -1 1
```

Локализация системы

Локализация консоли

```
[hostX:~] # sysinstall
(клавиатура, шрифты, терминал)

[hostX:~] # cat /etc/rc.conf
...
font8x16="cp866-8x16"
scrnmap="koi8-r2cp866"
keymap="ru.koi8-r"

[hostX:~] # /etc/rc.d/syscons start


[hostX:~] # rcsdiff /etc/ttys
35c35
< ttyv0 "/usr/libexec/getty Pc"         cons25  on  secure
---
> ttyv0 "/usr/libexec/getty Pc"         cons25r on  secure

[hostX:~] # kill -1 1

[hostX:~] # date

[hostX:~] # setenv LANG ru_RU.KOI8-R

[hostX:~] # date

[hostX:~] # pw usermod root -L russian
```

Локализация X сервера

Устанавливаем русскую локализацию пользователя

```
[hostX:~] # pw usermod root -L russian
[hostX:~] # logout
```

Включаем поддержку русских шрифтов

Штатные шрифты

```
[hostX:~] # cat xorg.conf
...
Section "Files"
        RgbPath      "/usr/X11R6/lib/X11/rgb"
        ModulePath   "/usr/X11R6/lib/modules"
```

```
          FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/"
        FontPath     "/usr/X11R6/lib/X11/fonts/misc/"
        FontPath     "/usr/X11R6/lib/X11/fonts/TTF/"
        FontPath     "/usr/X11R6/lib/X11/fonts/Type1/"
        FontPath     "/usr/X11R6/lib/X11/fonts/CID/"
        FontPath     "/usr/X11R6/lib/X11/fonts/75dpi/"
        FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection
...
```

TrueType шрифты

```
[hostX:~] # fetch http://koi8.pp.ru/dist/msttcorefonts.tgz

[hostX:~] # cd /usr/X11R6/lib/X11/fonts/
[hostX:lib/X11/fonts] # tar -xvzf ~/msttcorefonts.tgz

# cat /etc/X11/xorg.conf
Section "Files"
        RgbPath      "/usr/X11R6/lib/X11/rgb"
        ModulePath   "/usr/X11R6/lib/modules"
          FontPath     "/usr/X11R6/lib/X11/fonts/msttcorefonts/"
        FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/"
        FontPath     "/usr/X11R6/lib/X11/fonts/misc/"
        FontPath     "/usr/X11R6/lib/X11/fonts/TTF/"
        FontPath     "/usr/X11R6/lib/X11/fonts/Type1/"
        FontPath     "/usr/X11R6/lib/X11/fonts/CID/"
        FontPath     "/usr/X11R6/lib/X11/fonts/75dpi/"
        FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection
```

Включаем поддержку русской клавиатуры

На лету:

```
[hostX:~] # setxkbmap  -layout "us,ru" -option
"grp:alt_shift_toggle"
```

Или в файле конфигурации:

```
[hostX:~] # cat xorg.conf
...
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
          Option "XkbLayout" "us,ru"
          Option "XkbOptions" "grp:alt_shift_toggle"
EndSection
...
```

Управление оконными менеджерами

Инсталяция оконного менеджера

```
[gX:~] # pkg_add -r xfce4
```

Управление оконными менеджерами в скрипте startx

```
[gX:~] # cat .xinitrc
/usr/local/bin/xfce4-session
```

Управление оконными менеджерами в менеджере дисплеев xdm

```
[gX:~] # cat ~uX/.xsession
/usr/local/bin/xfce4-session
```

Работа в сетях Windows

Подключение к Microsoft терминал серверу

```
[gX:~] # pkg_add /usr/ports/packages/All/rdesktop.tbz
[gX:~] # rehash
[gX:~] # rdesktop 10.10.106.20
```

Монтирование файловых ресурсов Microsoft

```
root@shanti:~# mount -t cifs //195.19.32.31/IPTVMedia /mnt/ -o
user=Administrator
Password:
root@shanti:~# ls /mnt/
```

Работа с Internet сервисами

Сервис http

```
[gX:~] # pkg_add /usr/ports/packages/All/opera.tbz
```

Сервисы электронной почты

Сервисы сообщений

P2P сервисы

Работа с офисными документами

Работа с текстом

```
[gX:~] # pkg_add /usr/ports/packages/All/abiword.tbz
```

Сервис проверки орфографии

```
[gX:~] # pkg_add /usr/ports/packages/All/ru-aspell-0.60.2.tbz
```

Работа с мультимедийной информацией

Воспроизведение звука

Определяем драйвер звуковой карты

```
[gX:~] # kldload snd_driver

[gX:~] # cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH5 (82801EB)> at io 0xfc001000, 0xfc002000 irq 17
bufsz 16384 kld snd_ich (1p/1r/1v channels duplex default)
```

Прописываем его в автозагрузку

```
[gX:~] # cat /boot/loader.conf
...
snd_ich_load="YES"
...
```

Проверка работоспособности звуковой карты

```
[gX:~] # cat /usr/ports/addins/bird_1.au > /dev/audio0.0
```

Установка программы для воспроизведения аудиофайлов

```
[gX:~] # pkg_add /usr/ports/packages/All/mpg123.tbz

[gX:~] # rehash
[gX:~] # mpg123 /usr/ports/addins/KnockinonHeavensDoor.mp3
```

Воспроизведение видео

```
[gX:~] # pkg_add /usr/ports/packages/All/mplayer.tbz

[gX:~] # rehash
[gX:~] # mplayer -fs -zoom /usr/ports/addins/egik_v_tumane.avi
```

Система печати FreeBSD

Использование локального принтера

Рассматриваются принтеры HP
С некоторыми lpt принтерами приходилось:

```
# hw.intr_storm_threshold: 25000
# lptcontrol -s -d /dev/lpt0
```

Использование lpd

```
[gX:~] # pkg_add /usr/ports/packages/All/ghostscript-gpl-nox11.tbz

[gX:~] # cat /etc/printcap
lp|local line printer:\
        :sh:\
        :lp=/dev/ulpt0:\
        :sd=/var/spool/output/lpd:\
        :if=/usr/share/examples/printing/ifhp:\
        :lf=/var/log/lpd-errs:

[gX:~] # chmod +x /usr/share/examples/printing/ifhp

[gX:~] # cat >> /etc/rc.conf
lpd_enable="YES"

[gX:~] # /etc/rc.d/lpd start
Starting lpd.

[gX:~] # lpr /etc/rc.conf

[gX:~] # gunzip -c /usr/share/man/man1/cat.1.gz | groff -man -Tps
| lpr

[gX:~] # zcat /usr/share/man/man1/cat.1.gz | groff -man -Tlj4 |
lpr
```

Печать из Unix на Unix принт-сервере

Настройка сервера печати

Фильтр печати предполагается на клиенте

```
[gZ:~] # cat /etc/hosts.lpd
...
gX
...
```

```
[gZ:~] # cat /etc/printcap
lp|local line printer:\
        :sh:\
        :lp=/dev/ulpt0:\
        :sd=/var/spool/output/lpd:\
        :lf=/var/log/lpd-errs:
```

Настройка клиента

```
[gX:~] # cat /etc/printcap
lp|local line printer:\
        :sh:\
        :rm=gZ:\
        :sd=/var/spool/output/lpd:\
        :lf=/var/log/lpd-errs:\
        :if=/usr/share/examples/printing/ifhp:

[gX:~] # chmod +x /usr/share/examples/printing/ifhp

[gX:~] # cat /etc/rc.conf
...
lpd_enable="YES"
...

[gX:~] # /etc/rc.d/lpd start
Starting lpd.

[gX:~] # lpr /etc/rc.conf
```

Печать из Windows на Unix принт-сервере

Настройка сервера печати

Добавляем к конфигурации сервера печати:

```
[gX:~] # cat /usr/local/etc/smb.conf
[global]
    ...
        printcap name = /etc/printcap
        printing = bsd
    ...
[lp]
        path = /tmp
        printable = Yes
        print command = lpr -r %s
        use client driver = Yes
```

Печать из Unix на Windows принт-сервере

```
[gX:~] # pkg_info | grep 'cups\|samba\|ghost'
cups-base-1.3.9_2   Common UNIX Printing System
cups-pstoraster-8.15.4_2 Postscript interpreter for CUPS printing
to non-PS printers
cups-smb-backend-1.0_2 A CUPS backend for printing to Windows
servers
ghostscript8-nox11-8.62_5 Ghostscript 8.x PostScript interpreter
gutenprint-cups-5.1.7_3 GutenPrint Printer Driver
samba-3.0.32_2,1    A free SMB and CIFS client and server for UNIX

[gX:~] # cat /etc/rc.conf
...
cupsd_enable="YES"
...
```

Далее все через web интерфейс http://localhost:631

Использование эмуляторов

QEMU

```
[garant:~/qemu] % cat start_freebsd.sh
#!/bin/sh

# qemu-img create qemu_freebsd.img 512M
# qemu -cdrom /dev/acd0 -boot d qemu_freebsd.img &
# qemu -k ru -net tap,script=tun_up_freebsd.sh -net nic
qemu_freebsd.img &
qemu -net tap,script=tun_up_freebsd.sh -net nic qemu_freebsd.img &

/etc/rc.d/named stop
sleep 3
/etc/rc.d/named start

[garant:~/qemu] % cat tun_up_freebsd.sh
#!/bin/sh

ifconfig $1 192.168.113.1 netmask 255.255.255.0
```